DABC Generic Java GUI programming

H.G.Essel

January 8, 2009

Contents

Chapter 1

Package xgui

Package Contents *Page*

1.1 Interfaces

1.1.1 Interface xiDesktop

Interface to desktop. External components can let the xDesktop open/close frames (JInternalFrame). Interface is passed as argument in init function of xiUserPanel.

Declaration

```
public interface xiDesktop
```

Methods

- *addFrame*
  `public void` **addFrame**`( javax.swing.JInternalFrame` **frame** `)`
  - **Usage**
    - Adds a frame to desktop if a frame with same title does not exist.
  - **Parameters**
    - `frame` - Frame to put on desktop. Frame will be managed.

- *addFrame*
  `public void` **addFrame**`( javax.swing.JInternalFrame` **frame**`, boolean` **manage** `)`
  - **Usage**
    - Adds a frame to desktop if a frame with same title does not exist. Managed frames store/retreive their layout like GUI frames.
  - **Parameters**
    - `frame` - Frame to put on desktop.
    - `manage` - If true, frame will be managed by GUI: layout is saved and restored.

- *findFrame*
  `public boolean` **findFrame**`( java.lang.String` **title** `)`
  - **Usage**
    - Checks if a frame exists on the desktop.
  - **Parameters**
    - `title` - Title of the frame to searched for.
  - **Returns** - true if frame with specified title exists, or false.

- *removeFrame*
  `public void` **removeFrame**`( java.lang.String` **title** `)`
  - **Usage**
    - Remove (dispose) a frame from the desktop and list of managed frames.
  - **Parameters**

    * `title` - Title of the frame to be removed.

- *setFrameSelected*
  `public void` **setFrameSelected**`( java.lang.String` **title**`, boolean` **select** `)`
  - **Usage**
    * Switch a frames selection state (setSelected).
  - **Parameters**
    * `title` - Title of the frame to be selected.
    * `select` - passed to setSelected method of frame.

- *toFront*
  `public void` **toFront**`( java.lang.String` **title** `)`
  - **Usage**
    * Set frames to front.
  - **Parameters**
    * `title` - Title of the frame.

1.1.2 Interface **xiDimBrowser**

Interface of DIM browser.

Declaration

public interface xiDimBrowser

Methods

- *addInfoHandler*
  `public void` **addInfoHandler**`( xgui.xiDimParameter` **parameter**`,`
  `xgui.xiUserInfoHandler` **infohandler** `)`
  - **Usage**
    * Called in setDimServices of application panel to attach an info handler to a parameter. From the list returned by getParameters each to be handled by the panels handler must be added.
  - **Parameters**
    * `parameter` - Interface to parameter
    * `infohandler` - Interface of user info handler (application panel implementing xiUserInfohandler).

- *getCommands*
  `public Vector` **getCommands**`( )`
  - **Usage**
    * Called in setDimServices of application panel to get available commands.

  - **Returns** - Vector of command objects.

- *getParameters*
  `public Vector` **getParameters**`( )`
  - **Usage**
    - Called in setDimServices of application panel to get available parameters.
  - **Returns** - Vector of parameter objects.

- *removeInfoHandler*
  `public void` **removeInfoHandler**`( xgui.xiDimParameter` **parameter**`, xgui.xiUserInfoHandler` **infohandler** `)`
  - **Usage**
    - An info handler previously added to a parameter is removed (reference only).
  - **Parameters**
    - `parameter` - Interface to parameter
    - `infohandler` - Interface of user info handler (application panel implementing xiUserInfohandler).

- *sleep*
  `public void` **sleep**`( int` **seconds** `)`
  - **Usage**
    - Sleep some seconds.
  - **Parameters**
    - `seconds` -

1.1.3 Interface **xiDimCommand**

Interface to command objects.

Declaration

public interface xiDimCommand

Methods

- *exec*
  `public void` **exec**`( java.lang.String` **argument** `)`
  - **Usage**
    - Execute DIM command from internal parameter string (not from commandstring which is used only for sorting). If the application name is $:0 this was no DABC formatted command like DIM server EXIT and is handled differently.
  - **Parameters**

    * `argument` - String for command argument

      Note: If the command expects an integer or float argument, the string must be formatted properly.

- *getParserInfo*
  `public xiParser` **`getParserInfo( )`**
  - **Usage**
    * Get parser interface (keeps definitions and values).

      This interface provides only getter functions. It is called from external classes.
  - **Returns** - interface to parser provides access to all name fields.

1.1.4 Interface **xiDimParameter**

Interface to parameter objects. Mainly getter methods to access to parameter values.

Declaration

```
public interface xiDimParameter
```

Methods

- *getDoubleValue*
  `public double` **`getDoubleValue( )`**
- *getFloatValue*
  `public float` **`getFloatValue( )`**
- *getHisto*
  `public xRecordHisto` **`getHisto( )`**
- *getInfo*
  `public xRecordInfo` **`getInfo( )`**
- *getIntValue*
  `public int` **`getIntValue( )`**
- *getLongValue*
  `public long` **`getLongValue( )`**
- *getMeter*
  `public xRecordMeter` **`getMeter( )`**
- *getParserInfo*
  `public xiParser` **`getParserInfo( )`**
  - **Returns** - interface to parser provides access to all name fields.

- *getState*
  `public xRecordState` **`getState( )`**
- *getValue*
  `public String` **`getValue( )`**
- *setParameter*
  `public boolean` **`setParameter(`** `java.lang.String` **`value )`**
  - **Usage**
    - Builds and executes a DIM command **SetParameter name=vale** where name is the name part of the full DIM name string. The command **SetParameter** of cause must be implemented on the DIM server side. Called in xPanelParameter when a value is changed in the table.
  - **Parameters**
    - `value` -
  - **Returns** - completion status.

1.1.5 Interface xiPanelGraphics

Interface for JPanels to be put in xPanelGraphics.

Declaration

public interface xiPanelGraphics

Methods

- *getName*
  `public String` **`getName( )`**
- *setID*
  `public void` **`setID(`** `int` **`id )`**
- *setSizeXY*
  `public void` **`setSizeXY( )`**
- *setSizeXY*
  `public void` **`setSizeXY(`** `java.awt.Dimension` **`d )`**

1.1.6 Interface xiPanelItem

JPanel items to be placed into xPanelGraphics.

Declaration

public interface xiPanelItem

METHODS

- *getDimension*
  `public Dimension` **getDimension**`( )`
  - **Returns** - Current dimension (size).

- *getID*
  `public int` **getID**`( )`
  - **Returns** - ID unique in PanelGraphics (index).

- *getName*
  `public String` **getName**`( )`
  - **Returns** - Characteristic string of item.

- *getPanel*
  `public JPanel` **getPanel**`( )`
  - **Returns** - The panel of the class implementing the Interface. This panel is displayed in the PanelDisplay.

- *getPosition*
  `public Point` **getPosition**`( )`
  - **Returns** - Current position (relative to frame).

- *setActionListener*
  `public void` **setActionListener**`( java.awt.event.ActionListener` **actionlistener** `)`
  - **Parameters**
    - `actionlistener` - Optional actionlistener. If set, the action events from the panel item are passed through to this action listener.

- *setID*
  `public void` **setID**`( int` **id** `)`
  - **Usage**
    - Set internal ID.
  - **Parameters**
    - `id` - ID unique in PanelGraphics (index).

- *setSizeXY*
  `public void` **setSizeXY**`( )`
  - **Usage**
    - Sets the preferred size of item to internal vale.

- *setSizeXY*
  `public void` **setSizeXY**`( java.awt.Dimension` **d** `)`

- **Usage**
  - Sets the preferred size of item to specified dimension. Some items may resize all elements.
- **Parameters**
  - `d` - Dimension.

1.1.7 Interface **xiParser**

Interface to name parser. Format of names is:

Dns/Node:NodeId/applNS::Application:ApplicationID/Name.component

Declaration

```
public interface xiParser
```

Methods

- *getApplication*
  `public String` **getApplication**`( )`
- *getApplicationFull*
  `public String` **getApplicationFull**`( )`
- *getApplicationID*
  `public String` **getApplicationID**`( )`
- *getApplicationName*
  `public String` **getApplicationName**`( )`
- *getCommand*
  `public String` **getCommand**`( )`
- *getDns*
  `public String` **getDns**`( )`
- *getFormat*
  `public String` **getFormat**`( )`
- *getFull*
  `public String` **getFull**`( )`
- *getItems*
  `public String` **getItems**`( )`
- *getMode*
  `public int` **getMode**`( )`
- *getName*
  `public String` **getName**`( )`

- *getNameSpace*
  `public String` **getNameSpace**`( )`
- *getNode*
  `public String` **getNode**`( )`
- *getNodeID*
  `public String` **getNodeID**`( )`
- *getNodeName*
  `public String` **getNodeName**`( )`
- *getNofTypes*
  `public int` **getNofTypes**`( )`
- *getQuality*
  `public int` **getQuality**`( )`
- *getState*
  `public int` **getState**`( )`
- *getType*
  `public int` **getType**`( )`
- *getTypeList*
  `public String` **getTypeList**`( )`
- *getTypeSizes*
  `public int` **getTypeSizes**`( )`
- *getVisibility*
  `public int` **getVisibility**`( )`
- *isArray*
  `public boolean` **isArray**`( )`
- *isAtomic*
  `public boolean` **isAtomic**`( )`
- *isChangable*
  `public boolean` **isChangable**`( )`
- *isChar*
  `public boolean` **isChar**`( )`
- *isCommandDescriptor*
  `public boolean` **isCommandDescriptor**`( )`
- *isDouble*
  `public boolean` **isDouble**`( )`
- *isError*
  `public boolean` **isError**`( )`
- *isFatal*
  `public boolean` **isFatal**`( )`
- *isFloat*
  `public boolean` **isFloat**`( )`

- *isGeneric*
  `public boolean` **isGeneric**`( )`
- *isHidden*
  `public boolean` **isHidden**`( )`
- *isHistogram*
  `public boolean` **isHistogram**`( )`
- *isImportant*
  `public boolean` **isImportant**`( )`
- *isInfo*
  `public boolean` **isInfo**`( )`
- *isInformation*
  `public boolean` **isInformation**`( )`
- *isInt*
  `public boolean` **isInt**`( )`
- *isLogging*
  `public boolean` **isLogging**`( )`
- *isLong*
  `public boolean` **isLong**`( )`
- *isMonitor*
  `public boolean` **isMonitor**`( )`
- *isNotSpecified*
  `public boolean` **isNotSpecified**`( )`
- *isRate*
  `public boolean` **isRate**`( )`
- *isState*
  `public boolean` **isState**`( )`
- *isStruct*
  `public boolean` **isStruct**`( )`
- *isSuccess*
  `public boolean` **isSuccess**`( )`
- *isVisible*
  `public boolean` **isVisible**`( )`
- *isWarning*
  `public boolean` **isWarning**`( )`

1.1.8 Interface **xiUserCommand**

Interface to be implemented by application panels. Application panels are JPanels.

DECLARATION

public interface xiUserCommand

METHODS

- *getArgumentStyleXml*
  `public boolean` **getArgumentStyleXml**`( java.lang.String` **scope**`, java.lang.String` **command** `)`
  - **Usage**
    - Called by xPanelCommand.
  - **Parameters**
    - `scope` - of command
    - `command` - string
  - **Returns** - true, if command should be XML coded, false otherwise (MBS).
  - **See Also**
    - `xgui.xPanelCommand` (in 1.2.19, page 54)

1.1.9 INTERFACE xiUserInfoHandler

Interface to be implemented by application panels. Application panels are JPanels. In setDimServices (xiUserPanel) this interface must be attached to a DIM parameter by browser function addInfoHandler.

DECLARATION

public interface xiUserInfoHandler

METHODS

- *getName*
  `public String` **getName**`( )`
  - **Usage**
    - Called in callback of DIM parameter.
  - **Returns** - Unique name of handler.

- *infoHandler*
  `public void` **infoHandler**`( xgui.xiDimParameter` **parameter** `)`
  - **Usage**
    - Called in callback of DIM parameter.
  - **Parameters**
    - `parameter` - Interface to parameter which has been changed.
  - **See Also**
    - `xgui.xiDimParameter` (in 1.1.4, page 8)

1.1.10 Interface **xiUserPanel**

Interface to be implemented by application panels. Application panels are JPanels.

Declaration

```
public interface xiUserPanel
```

Methods

- *getHeader*
  `public String` **getHeader**`( )`
  - **Usage**
    - Called by xDesktop.
  - **Returns** - String to be used as header.

- *getIcon*
  `public ImageIcon` **getIcon**`( )`
  - **Usage**
    - Called by xDesktop.
  - **Returns** - ImageIcon for start button.

- *getToolTip*
  `public String` **getToolTip**`( )`
  - **Usage**
    - Called by xDesktop.
  - **Returns** - String for tool tip.

- *getUserCommand*
  `public xiUserCommand` **getUserCommand**`( )`
  - **Usage**
    - Called by xDesktop.
  - **Returns** - Object implementing xiUserCommand

- *init*
  `public void` **init**`( xgui.xiDesktop` **desktop**`, java.awt.event.ActionListener` **actionlistener** `)`
  - **Usage**
    - Called by xDesktop.
  - **Parameters**
    - `desktop` - Interface to desktop.

    * `actionlistener` - The action listener of the desktop. Events may be passed to this. Not sure if this parameter will remain, because it is dangerous to allow the application to pass events to the desktops listener.

- *releaseDimServices*
  `public void` **releaseDimServices**`( )`
  - **Usage**
    * Called from desktop after every change in DIM services. Application panel must release local references to DIM parameters and commands.

- *setDimServices*
  `public void` **setDimServices**`( xgui.xiDimBrowser` **browser** `)`
  - **Usage**
    * Called from desktop after releaseDimServices() after every change in DIM services. Application panel must build all references to DIM services.
  - **Parameters**
    * `browser` - Interface to browser.

1.2 Classes

1.2.1 Class xConvert

Swapping function.

Declaration

public class xConvert **extends** java.lang.Object

Constructors

- *xConvert*
  `public` **xConvert**`( )`

Methods

- *istr*
  `public static final String` **istr**`( java.io.DataInputStream` **in**`, int` **e** `)`
- *iswap*
  `public static final int` **iswap**`( java.io.DataInputStream` **in**`, int` **e** `)`
- *str*
  `public static final String` **str**`( java.io.DataInputStream` **in**`, byte [] b )`
- *swap*
  `public static final int` **swap**`( int` **i**`, int` **e** `)`

1.2.2 Class **xCrypt**

Encrypt passwords.

Declaration

public class xCrypt
extends java.lang.Object

Methods

- *crypt*
  `public static final String` **crypt**`( java.lang.String` **original** `)`
  - **Usage**
    *

      Encrypt a password given the cleartext password. This method generates a random salt using the 'java.util.Random' class.
  - **Parameters**
    * `original` - The password to be encrypted.
  - **Returns** - A string consisting of the 2-character salt followed by the encrypted password.

- *crypt*
  `public static final String` **crypt**`( java.lang.String` **salt**`, java.lang.String` **original** `)`
  - **Usage**
    *

      Encrypt a password given the cleartext password and a "salt".
  - **Parameters**
    * `salt` - A two-character string representing the salt used to iterate the encryption engine in lots of different ways. If you are generating a new encryption then this value should be randomised.
    * `original` - The password to be encrypted.
  - **Returns** - A string consisting of the 2-character salt followed by the encrypted password.

- *matches*
  `public static final boolean` **matches**`( java.lang.String` **encryptedPassword**`, java.lang.String` **enteredPassword** `)`
  - **Usage**
    *

      Check that *enteredPassword* encrypts to *encryptedPassword*.

- **Parameters**
  - `encryptedPassword` - The *encryptedPassword*. The first two characters are assumed to be the salt. This string would be the same as one found in a Unix <U>/etc/passwd</U>file.
  - `enteredPassword` - The password as entered by the user (or otherwise aquired).
- **Returns** - **true** if the password should be considered correct.

1.2.3 Class **xDesktop**

Top desktop class. This object manages the main window including the main toolbar and a JDesktopPane in which all windows (JPanel) are opened as JInternalFrame. It also creates the DIM browser object which manages the list of parameters and commands.

Declaration

public class xDesktop
extends javax.swing.JFrame
implements xiDesktop, java.awt.event.ActionListener

Serializable Fields

Constructors

- *xDesktop*
  `public` **xDesktop**`( xgui.xiUserPanel` **userpanel**`, boolean` **control** `)`
  - **Usage**
    - Creates the top level GUI.
  - **Parameters**
    - `userpanel` - Optional user panel. If null, user panel class name could alternatively be specified as DABC_USER_PANEL and will be instantiated.
    - `control` - If false, no control panels are opened.

Methods

- *actionPerformed*
  `public void` **actionPerformed**`( java.awt.event.ActionEvent` **e** `)`
  - **Usage**
    - Central action switch.

· Update

releaseDimServices() of all form panels.

Browser releaseServices (deactivate all parameters and remove user handlers).

Browser initServices (get new list of services). Merge new ones into existing list.

Create new parameter panel (Wait for update of all parameters, cleanup graphics panels, build new table, mark all parameters as not shown to rebuild graphics, build new command definition list).

Create new command panel (build command tree)

updateAll graphical panels.

setDimServices of all form panels

enableServices (activate all parameters)

Pass command descriptors to command panel

Replace parameter, command, and info panel in their window frames.

– **Parameters**
  * `e` - event. Switch on action command:

- *addFrame*
  `public void` **addFrame**`( javax.swing.JInternalFrame` **frame** `)`
  – **Usage**
    * Adds a frame to desktop if a frame with same title does not exist.
  – **Parameters**
    * `frame` - Frame to put on desktop. Frame will be managed.

- *addFrame*
  `public void` **addFrame**`( javax.swing.JInternalFrame` **frame**`, boolean` **manage** `)`
  – **Usage**
    * Adds a frame to desktop.
  – **Parameters**
    * `frame` - Frame to put on desktop.
    * `manage` - If true, frame will be managed by GUI: layout is saved and restored.

- *findFrame*
  `public boolean` **findFrame**`( java.lang.String` **title** `)`
  – **Usage**

    * Checks if a frame exists on the desktop .
  - **Parameters**
    * `title` - Title of the frame to searched for.
  - **Returns** - true if frame with specified title exists, or false.

- *quit*
  `protected void` **quit**`( )`

- *removeFrame*
  `public void` **removeFrame**`( java.lang.String` **title** `)`
  - **Usage**
    * Remove (dispose) a frame from the desktop and list of managed frames.
  - **Parameters**
    * `title` - Title of the frame to be removed.

- *setFrameSelected*
  `public void` **setFrameSelected**`( java.lang.String` **title**`, boolean` **select** `)`
  - **Usage**
    * Switch a frames selection state (setSelected).
  - **Parameters**
    * `title` - Title of the frame to be selected.
    * `select` - passed to setSelected method of frame.

- *toFront*
  `public void` **toFront**`( java.lang.String` **title** `)`
  - **Usage**
    * Set frames to front.
  - **Parameters**
    * `title` - Title of the frame.

1.2.4 Class **xDimBrowser**

DIM browser.

Declaration

public class xDimBrowser
extends java.lang.Object
implements xiDimBrowser

Constructors

- *xDimBrowser*
  `public` **xDimBrowser**`( xgui.xPanelHisto` **histogram**`, xgui.xPanelMeter` **meter**`, xgui.xPanelState` **state**`, xgui.xPanelInfo` **info** `)`
  - **Usage**
    - Constructor adds DIM error handler.
  - **Parameters**
    - `histogram` - Panel
    - `meter` - Panel
    - `state` - Panel
    - `info` - Panel

Methods

- *addInfoHandler*
  `public void` **addInfoHandler**`( xgui.xiDimParameter` **parameter**`, xgui.xiUserInfoHandler` **infohandler** `)`
- *enableServices*
  `protected void` **enableServices**`( )`
  - **Usage**
    - Steps through parameter list and activates all by setParameterActiv.
  - **See Also**
    - `xgui.xDimParameter` (in 1.2.7, page 25)
- *getCommandList*
  `protected Vector` **getCommandList**`( )`
- *getCommands*
  `public Vector` **getCommands**`( )`
- *getNofServers*
  `protected int` **getNofServers**`( )`
  - **Returns** - Number of servers (number of EXIT commands).
- *getNumberOfCommands*
  `protected int` **getNumberOfCommands**`( )`
- *getNumberOfParameters*
  `protected int` **getNumberOfParameters**`( )`
- *getPanelHistogram*
  `protected xPanelHisto` **getPanelHistogram**`( )`
- *getPanelInfo*
  `protected xPanelInfo` **getPanelInfo**`( )`
- *getPanelMeter*
  `protected xPanelMeter` **getPanelMeter**`( )`

- *getPanelState*
  `protected xPanelState` **getPanelState**`( )`

- *getParameterList*
  `protected Vector` **getParameterList**`( )`

- *getParameters*
  `public Vector` **getParameters**`( )`

- *getServers*
  `protected String` **getServers**`( )`
  - **Usage**
    - Search for server node names. Searches for * / EXIT services.
  - **Returns** - Space separated list of servres names.

- *getServices*
  `protected String` **getServices**`( java.lang.String` **wildcard** `)`
  - **Usage**
    - Get list of services filtered by wildcard.
  - **Parameters**
    - `wildcard` - Wildcard string
  - **Returns** - String array of service names.

- *initServices*
  `protected void` **initServices**`( java.lang.String` **wildcard** `)`
  - **Usage**
    - Get list of services from DIM name server defined by DIM_DNS_NODE, filtered by wildcard. Commands and parameters are kept in separate lists and ordered alphabetically. Names not matching the DABC format (four elements separated by slashes) are handled separately. Only DIM EXIT commands are formatted according DABC, but in xDimCommand they are restored for execution. This might need better solution.
  - **Parameters**
    - `wildcard` - services name filter
  - **See Also**
    - `xgui.xParser` (in 1.2.33, page 76)

- *listServices*
  `protected void` **listServices**`( boolean` **all** `)`

- *releaseServices*
  `protected void` **releaseServices**`( boolean` **cleanup** `)`
  - **Parameters**
    - `cleanup` - True: Remove all services, otherwise only deactivate.
  - **See Also**
    - `xgui.xDimParameter` (in 1.2.7, page 25)

- *removeInfoHandler*
  `public void` **removeInfoHandler**`( xgui.xiDimParameter` **parameter**`, xgui.xiUserInfoHandler` **infohandler** `)`
- *sleep*
  `public void` **sleep**`( int` **s** `)`
- *startTimer*
  `public static void` **startTimer**`( int` **secs** `)`
  - **Usage**
    - Timer

1.2.5 Class **xDimCommand**

DIM command class

Declaration

public class xDimCommand
extends java.lang.Object
implements xiDimCommand

Constructors

- *xDimCommand*
  `public` **xDimCommand**`( java.lang.String` **name**`, java.lang.String` **format**`, int` **version** `)`
  - **Usage**
    - Create DIM command object
  - **Parameters**
    - `name` - DABC format full command string. String and format are parsed and stored in parser.
    - `format` - DIM format list
    - `version` - number of instance (debug purpose only)

Methods

- *exec*
  `public void` **exec**`( java.lang.String` **arg** `)`
- *getParser*
  `protected xParser` **getParser**`( )`
  - **Usage**
    - Get parser keeping fields and formats ov DIM command.
  - **Returns** - parser

- *getParserInfo*
  `public xiParser` **getParserInfo**`( )`

- *getType*
  `protected String` **getType**`( )`
  - **Usage**
    - Used by xPanelCommand.
  - **Returns** - data type

- *getXmlParser*
  `protected xXmlParser` **getXmlParser**`( )`
  - **Usage**
    - Get XML parser (keeps definitions and values)
  - **Returns** - XML parser

- *setIndent*
  `protected void` **setIndent**`( int` **ind** `)`
  - **Usage**
    - Set indentation level. This controls which field is returned by toString (used by tree browser). In the browser the order goes from command to application to node.
  - **Parameters**
    - `ind` - indent level for tree browser.

- *setXmlParser*
  `protected void` **setXmlParser**`( xgui.xXmlParser` **parser** `)`
  - **Usage**
    - Specify XML parser to be used (keeps definitions and values)
  - **Parameters**
    - `parser` - XML parser

- *toString*
  `public String` **toString**`( )`
  - **Returns** - field with last indentation

- *toString*
  `public String` **toString**`( int` **ind** `)`
  - **Parameters**
    - `ind` - indent level for tree browser.
  - **Returns** - field according indentation and store indentation used in toString()

1.2.6 Class **xDimNameInfo**

InfoHandler to manage list of DIM servers. List is composed in a text area. DIM parameter is DIS_DNS/SERVER_LIST.

DECLARATION

public class xDimNameInfo
extends dim.DimInfo

CONSTRUCTORS

- *xDimNameInfo*
  `public` **xDimNameInfo**`( java.lang.String` **service**`, javax.swing.JTextArea` **label** `)`
  - **Usage**
    - Constructor of DIM parameter handler.
  - **Parameters**
    - `service` - DIM name of service: DIS_DNS/SERVER_LIST
    - `label` - Text area to store the DIM server list.

METHODS

- *infoHandler*
  `public void` **infoHandler**`( )`
  - **Usage**
    - The DIM parameter DIS_DNS/SERVER_LIST is either a list, or incremental. That means it may start with + to add a server, or with - to remove a server, or a list of servers separated by —:

      +name@node -name@node name@node—name@node—name@node

      This handler only handles the increments. On startup the text area is filled with the server list by xBrowser.getServers function.
  - **See Also**
    - `xgui.xDimBrowser` (in 1.2.4, page 20)

1.2.7 CLASS xDimParameter

A list of these objects is the central management of DIM parameters. It implements the DIM handler, creates the graphical elements, the records keeping parameter values, and interface functions to access these values. It also has reference to table model. On parameter update, the table and all graphical objects are updated.

DECLARATION

public class xDimParameter
extends dim.DimInfo
implements xiDimParameter

Constructors

- *xDimParameter*
  `public` **xDimParameter**`( java.lang.String` **name**`, java.lang.String` **format**`, float` **noLink**`, int` **version** `)`
  - **Usage**
    - DIM float parameter. Calls initParser
  - **Parameters**
    - `name` - DABC format parameter name
    - `format` - DIM format list
    - `noLink` - default value if no connection to DIM server
    - `version` - instance number (for internal debugging only)

- *xDimParameter*
  `public` **xDimParameter**`( java.lang.String` **name**`, java.lang.String` **format**`, int` **noLink**`, int` **version** `)`
  - **Usage**
    - DIM integer parameter. Calls initParser
  - **Parameters**
    - `name` - DABC format full parameter name
    - `format` - DIM format list
    - `noLink` - default value if no connection to DIM server
    - `version` - instance number (for internal debugging only)

- *xDimParameter*
  `public` **xDimParameter**`( java.lang.String` **name**`, java.lang.String` **format**`, java.lang.String` **noLink**`, int` **version** `)`
  - **Usage**
    - DIM string parameter. Calls initParser
  - **Parameters**
    - `name` - DABC format parameter name
    - `format` - DIM format list
    - `noLink` - default value if no connection to DIM server
    - `version` - instance number (for internal debugging only)

Methods

- *addInfoHandler*
  `protected void` **addInfoHandler**`( xgui.xiUserInfoHandler` **pu** `)`
  - **Usage**
    - Add user handler. Called from user panels through xiDimBrowser interface.
  - **Parameters**
    - `pu` - Interface of user handler
  - **See Also**
    - `xgui.xiDimBrowser` (in 1.1.2, page 6)

- *addRow*
  `protected boolean` **addRow**`( xgui.xParTable` **table**`, int` **rowindex** `)`
  - **Usage**
    - Adds a new row to the table. Called in xPanelParameter.initPanel. Only visible parameters are handled. Graphical elements are created like meters, if they are monitored.
  - **Parameters**
    - `table` - Table model assigned to this parameter.
    - `rowindex` - index of row in table.
  - **Returns** - true parameter is visible and has been added, else parameter is not visible and has not been added.
  - **See Also**
    - `xgui.xPanelParameter` (in 1.2.28, page 67)

- *createHisto*
  `protected void` **createHisto**`( java.lang.Boolean` **create** `)`
  - **Parameters**
    - `create` - Create or remove histogram (to/from panel).

- *createInfo*
  `protected void` **createInfo**`( java.lang.Boolean` **create** `)`
  - **Parameters**
    - `create` - Create or remove info (to/from panel).

- *createMeter*
  `protected void` **createMeter**`( java.lang.Boolean` **create** `)`
  - **Parameters**
    - `create` - Create or remove meter (to/from panel).

- *createState*
  `protected void` **createState**`( java.lang.Boolean` **create** `)`
  - **Parameters**
    - `create` - Create or remove state (to/from panel).

- *getDimQuality*
  `public int` **getDimQuality**`( )`
  - **Returns** - quality.

- *getDoubleValue*
  `public double` **getDoubleValue**`( )`
  - **Returns** - value.

- *getFloatValue*
  `public float` **getFloatValue**`( )`
  - **Returns** - value.

- *getHisto*
  `public xRecordHisto` **getHisto**`( )`

- *getInfo*
  `public xRecordInfo` **getInfo**`( )`

- *getIntValue*
  `public int` **getIntValue**`( )`
  - **Returns** - value.

- *getLongValue*
  `public long` **getLongValue**`( )`
  - **Returns** - value.

- *getMeter*
  `public xRecordMeter` **getMeter**`( )`
  - **Usage**
    - Meter record is updated from meter settings.
  - **Returns** - Meter record.

- *getNode*
  `protected String` **getNode**`( )`
  - **Returns** - node:ID form parser.

- *getParser*
  `protected xParser` **getParser**`( )`
  - **Returns** - parser object.

- *getParserInfo*
  `public xiParser` **getParserInfo**`( )`
  - **Returns** - parser interface

- *getState*
  `public xRecordState` **getState**`( )`

- *getValue*
  `public String` **getValue**`( )`
  - **Returns** - value.

- *getXmlParser*
  `protected xXmlParser` **getXmlParser**`( )`
  - **Returns** - command parser object.

- *infoHandler*
  `public void` **infoHandler**`( )`
  - **Usage**

    * Info handler. Checks the incoming name and format against the stored ones. Table field and rate meter are updated, if known.

- *initParser*
  `protected void` **initParser(** `java.lang.String` **name,** `java.lang.String` **format** **)**
  - **Usage**
    * Initializes name parser. Creates XML parser. Creates command to set parameter value by preceding underscore to the parameter name. Value is set to string of NOLINK.
  - **Parameters**
    * `name` - DABC format parameter name
    * `format` - DIM format list

- *isCommandDescriptor*
  `protected boolean` **isCommandDescriptor(** **)**
  - **Returns** - true if command descriptor

- *printParameter*
  `public void` **printParameter(** `boolean` **comdef** **)**

- *printParameter*
  `public void` **printParameter(** `int` **index** **)**

- *removeInfoHandler*
  `protected void` **removeInfoHandler(** `xgui.xiUserInfoHandler` **pu** **)**
  - **Usage**
    * Remove user handler. Called from user panels through xiDimBrowser interface.
  - **Parameters**
    * `pu` - Interface of user handler
  - **See Also**
    * `xgui.xiDimBrowser` (in 1.1.2, page 6)

- *setAttributeHisto*
  `public void` **setAttributeHisto(** **)**

- *setAttributeMeter*
  `public void` **setAttributeMeter(** **)**

- *setIndex*
  `protected void` **setIndex(** `int` **index** **)**
  - **Usage**
    * Called by xPanelParameter after sorting
  - **Parameters**
    * `index` - parameter index shown as text in first column.

- *setPanels*
  `protected void` **setPanels(** `xgui.xPanelHisto` **histogramPanel,** `xgui.xPanelMeter` **meterPanel,** `xgui.xPanelState` **statePanel,** `xgui.xPanelInfo` **infoPanel** **)**

  - **Usage**
    * Imports the references to the graphic panels. Called from browser.
  - **Parameters**
    * `histogramPanel` -
    * `meterPanel` -
    * `statePanel` -
    * `infoPanel` -

- *setParameter*
  `public boolean` **setParameter**`( java.lang.String` **arg** `)`
  - **Usage**
    * Execute DIM command from internal parameter string .
  - **Parameters**
    * `arg` - string for command argument

- *setParameterActiv*
  `protected void` **setParameterActiv**`( boolean` **activ** `)`
  - **Usage**
    * (De)activate parameter. A deactivated parameter does no drawing in infoHandler function neither calls user handler. Before any changes in the parameter list is done, all parameters are deactivated (browser).
  - **Parameters**
    * `activ` - true: Activate and redraw all graphic objects, otherwise deactivate.

- *setPrint*
  `protected void` **setPrint**`( boolean` **dop** `)`

- *setTableIndex*
  `protected void` **setTableIndex**`( int` **index** `)`
  - **Usage**
    * Called by PanelParameter after sorting
  - **Parameters**
    * `index` - table index needed to update correct row.

- *toString*
  `public String` **toString**`( )`
  - **Usage**
    * The first row of the table is the DimParameter object. The string seen in the table is the string returned by this function.
  - **Returns** - index as string used for table.

1.2.8 Class **xForm**

Base class to keep the data of the setup forms for MBS and DABC

Declaration

public class xForm
extends java.lang.Object

Constructors

- *xForm*
  `public` **xForm**`( )`
- *xForm*
  `public` **xForm**`( java.awt.event.ActionListener` **a** `)`

Methods

- *addActionListener*
  `protected void` **addActionListener**`( java.awt.event.ActionListener` **ae** `)`
- *getActionListener*
  `public ActionListener` **getActionListener**`( )`
- *getLaunchFile*
  `public String` **getLaunchFile**`( )`
- *getMaster*
  `public String` **getMaster**`( )`
- *getScript*
  `public String` **getScript**`( )`
- *getServers*
  `public String` **getServers**`( )`
- *getSystemPath*
  `public String` **getSystemPath**`( )`
- *getUserPath*
  `public String` **getUserPath**`( )`
- *setLaunchFile*
  `protected void` **setLaunchFile**`( java.lang.String` **file** `)`
- *setMaster*
  `protected void` **setMaster**`( java.lang.String` **master** `)`
- *setScript*
  `protected void` **setScript**`( java.lang.String` **script** `)`
- *setServers*
  `protected void` **setServers**`( java.lang.String` **servers** `)`
- *setSystemPath*
  `protected void` **setSystemPath**`( java.lang.String` **systempath** `)`
- *setUserPath*
  `protected void` **setUserPath**`( java.lang.String` **userpath** `)`

1.2.9 Class **xFormDabc**

Base class to keep the data of the setup forms for DABC. Reads/writes XML setup file.

Declaration

public class xFormDabc
extends xgui.xFormDabc

Constructors

- *xFormDabc*
  `public` **xFormDabc**`( )`
- *xFormDabc*
  `public` **xFormDabc**`( java.lang.String` **file** `)`
- *xFormDabc*
  `public` **xFormDabc**`( java.lang.String` **file**`, java.awt.event.ActionListener` **action** `)`

Methods

- *getName*
  `public String` **getName**`( )`
  - **Returns** - DABC master name
- *getSetup*
  `public String` **getSetup**`( )`
  - **Returns** - DABC setup file name
- *printForm*
  `protected void` **printForm**`( )`
- *restoreSetup*
  `protected void` **restoreSetup**`( java.lang.String` **file** `)`
- *saveSetup*
  `protected void` **saveSetup**`( java.lang.String` **file** `)`
- *setDefaults*
  `protected void` **setDefaults**`( )`
- *setName*
  `protected void` **setName**`( java.lang.String` **name** `)`
  - **Parameters**
    - `name` - DABC master name

- *setSetup*
  `protected void` **setSetup**`( java.lang.String` **setup** `)`
  - **Parameters**
    - `setup` - DABC setup file name

1.2.10 Class xFormMbs

Base class to keep the data of the setup forms for MBS Reads/writes XML setup file.

Declaration

public class xFormMbs
extends xgui.xForm

Constructors

- *xFormMbs*
  `public` **xFormMbs**`( )`
- *xFormMbs*
  `public` **xFormMbs**`( java.lang.String` **file** `)`
- *xFormMbs*
  `public` **xFormMbs**`( java.lang.String` **file**`, java.awt.event.ActionListener` **action** `)`

Methods

- *getCommand*
  `public String` **getCommand**`( )`
- *printForm*
  `protected void` **printForm**`( )`
- *restoreSetup*
  `protected void` **restoreSetup**`( java.lang.String` **file** `)`
- *saveSetup*
  `protected void` **saveSetup**`( java.lang.String` **file** `)`
- *setCommand*
  `protected void` **setCommand**`( java.lang.String` **command** `)`
- *setDefaults*
  `protected void` **setDefaults**`( )`

1.2.11 Class **xGui**

Main class. Creates desktop.

Optionally a switch -m may be passed to indicate that no control panels will be shown (monitoring mode).

A class name (class must implement interface xiUserPanel) may optionally be specified which is instantiated and the object is passed as xiUserPanel to the created desktop.

Declaration

public class xGui
extends java.lang.Object

Constructors

- *xGui*
  `public` **xGui**`( )`

Methods

- *main*
  `public static void` **main**`( java.lang.String [] args )`
  - **Usage**
    - Main entry. Checks for DIM_DNS_NODE and application class argument, then starts event-dispatching thread. Sets default Locale to "en" and "US".
  - **Parameters**
    - `args` - optional -m for monitoring only or optional class name of user panel.

1.2.12 Class **xHisto**

Panel with histogram display.

Declaration

public class xHisto
extends javax.swing.JPanel
implements xiPanelItem, java.awt.event.MouseMotionListener, java.awt.event.MouseListener, java.awt.event.ActionListener, java.awt.event.ComponentListener

Serializable Fields

Fields

- public static final boolean LOG
  - Can be used in setLogScale or redraw.
- public static final boolean LIN
  - Can be used in setLogScale or redraw.
- public static final int LINE
  - Line mode drawing.
- public static final int BAR
  - Bar mode drawing.
- public static final int XSIZE
  - recommended size x
- public static final int YSIZE
  - recommended size y
- public static final int XSIZE_LARGE
  - recommended large size x
- public static final int YSIZE_LARGE
  - recommended large size y

Constructors

- *xHisto*
  `public` **xHisto**`( java.lang.String` **name**`, java.lang.String` **head**`, java.lang.String` **cont**`, java.lang.String` **xaxis**`, int` **x**`, int` **y** `)`
  - **Usage**
    - Create panel.
  - **Parameters**
    - `name` - Name of panel (returned by getName()).
    - `head` - Headline.
    - `cont` - Content lettering (Y-axis).
    - `xaxis` - Content lettering (X-axix).
    - `x` - Width.
    - `y` - Height.

METHODS

- *actionPerformed*
  `public void` **actionPerformed**`( java.awt.event.ActionEvent` **a** `)`
- *componentHidden*
  `public void` **componentHidden**`( java.awt.event.ComponentEvent` **e** `)`
- *componentMoved*
  `public void` **componentMoved**`( java.awt.event.ComponentEvent` **e** `)`
- *componentResized*
  `public void` **componentResized**`( java.awt.event.ComponentEvent` **e** `)`
- *componentShown*
  `public void` **componentShown**`( java.awt.event.ComponentEvent` **e** `)`
- *getClone*
  `public xHisto` **getClone**`( java.lang.String` **name** `)`
  - **Usage**
    - Create a new histogram from existing one. When clone shall be attached to parent histogram, typically these settings must be applied:

      clone.setSizeXY(new Dimension(XSIZE_LARGE,YSIZE_LARGE));

      clone.hasParent();

      parent.setExternHisto(clone,Frame);
  - **Parameters**
    - `name` - Name of clone.
- *getColor*
  `public String` **getColor**`( )`
- *getDimension*
  `public Dimension` **getDimension**`( )`
- *getID*
  `public int` **getID**`( )`
- *getLogScale*
  `public boolean` **getLogScale**`( )`
- *getMode*
  `public int` **getMode**`( )`
- *getName*
  `public String` **getName**`( )`
- *getPanel*
  `public JPanel` **getPanel**`( )`
- *getPosition*
  `public Point` **getPosition**`( )`

- *getSizeX*
  `public int` **getSizeX**`( )`

- *getSizeY*
  `public int` **getSizeY**`( )`

- *hasParent*
  `protected void` **hasParent**`( )`
  - **Usage**
    - If caller makes a clone of this Histogram (parent), and wants that clone to be controlled by this Histogram (updated), it must set the clone to hasParent. The clone is passed to its parent by setExternHisto together with a reference to a JInternalFrame where the clone is displayed. This frame must be created by caller. If frame is closed, the parent removes its child.

- *mouseClicked*
  `public void` **mouseClicked**`( java.awt.event.MouseEvent` **me** `)`

- *mouseDragged*
  `public void` **mouseDragged**`( java.awt.event.MouseEvent` **me** `)`

- *mouseEntered*
  `public void` **mouseEntered**`( java.awt.event.MouseEvent` **me** `)`

- *mouseExited*
  `public void` **mouseExited**`( java.awt.event.MouseEvent` **me** `)`

- *mouseMoved*
  `public void` **mouseMoved**`( java.awt.event.MouseEvent` **me** `)`

- *mousePressed*
  `public void` **mousePressed**`( java.awt.event.MouseEvent` **me** `)`
  - **Usage**
    - Context menu definition (RMB)

- *mouseReleased*
  `public void` **mouseReleased**`( java.awt.event.MouseEvent` **me** `)`

- *paintComponent*
  `public void` **paintComponent**`( java.awt.Graphics` **g** `)`
  - **Usage**
    - Called by repaint, calls update.

- *redraw*
  `public void` **redraw**`( )`
  - **Usage**
    - Final redraw. Calls repaint.

- *redraw*
  `public void` **redraw**`( int` **channels**`, int [] iBuffer, boolean` **draw** `)`
  - **Usage**

    * Redraw with new value.
  - **Parameters**
    * `channels` - Number of channels
    * `iBuffer` - Integer field with data.
    * `draw` - True: call redraw, otherwise no repaint.

- *redraw*
  `public void` **redraw**`( java.lang.String` **head**`, java.lang.String` **cont**`, java.lang.String` **xaxis**`, float []` **dBuffer**`, int` **channels**`, boolean` **log**`, java.awt.Color` **c** `)`

  - **Usage**
    * Final redraw. Calls repaint.
  - **Parameters**
    * `head` - Headline.
    * `cont` - Content lettering (Y-axis).
    * `xaxis` - Content lettering (X-axix).
    * `dBuffer` - Data field.
    * `channels` - Number of channels
    * `log` - Logarithmic scale?
    * `c` - Color.

- *setActionListener*
  `public void` **setActionListener**`( java.awt.event.ActionListener` **actionlistener** `)`

- *setBar*
  `public void` **setBar**`( boolean` **drawbar** `)`

- *setColor*
  `public void` **setColor**`( java.awt.Color` **c** `)`

- *setColor*
  `public void` **setColor**`( java.lang.String` **colorname** `)`

  - **Usage**
    * Set color by name.
  - **Parameters**
    * `colorname` - (Red, Green, Blue, Yellow, Cyan, Magenta).

- *setColorBack*
  `public void` **setColorBack**`( java.awt.Color` **color** `)`

- *setExternHisto*
  `protected void` **setExternHisto**`( xgui.xHisto` **exthisto**`, javax.swing.JInternalFrame` **extframe** `)`

  - **Usage**
    * Attach a second histogram to this. Second histogram will be updated from this.
  - **Parameters**
    * `exthisto` - Typically a clone with large size to be displayed in an extra frame.
    * `extframe` - Extra frame for the clone. Must be handled by caller.

- *setID*
  `public void` **setID**`( int` **i** `)`
- *setLettering*
  `public void` **setLettering**`( java.lang.String` **head**`, java.lang.String` **cont**`, java.lang.String` **xaxis** `)`
  - **Parameters**
    - `head` - Headline.
    - `cont` - Content lettering (Y-axis).
    - `xaxis` - Content lettering (X-axix).
- *setLogScale*
  `public void` **setLogScale**`( boolean` **log** `)`
- *setMode*
  `public void` **setMode**`( int` **Mode** `)`
- *setSizeXY*
  `public void` **setSizeXY**`( )`
- *setSizeXY*
  `public void` **setSizeXY**`( java.awt.Dimension` **dd** `)`
- *update*
  `public void` **update**`( java.awt.Graphics` **g** `)`
  - **Usage**
    - Overwriting update method we avoid clearing the graphics. This would cause flickering. Update is not called by repaint!

1.2.13 Class **xInfo**

Graphic item info line.

Declaration

public class xInfo
extends javax.swing.JPanel
implements xiPanelItem, java.awt.event.MouseListener, java.awt.event.ActionListener

Serializable Fields

FIELDS

- public static final int XSIZE
  - recommended size x
- public static final int YSIZE
  - recommended size y

CONSTRUCTORS

- *xInfo*
  `public` **xInfo**`( java.lang.String` **head**`, int` **xlength**`, int` **ylength** `)`
  - **Usage**
    - Creates a Info canvas.
  - **Parameters**
    - `head` - Name of parameter displayed.
    - `xlength` - Size of canvas in pixels.
    - `ylength` - Size of canvas in pixels.

METHODS

- *actionPerformed*
  `public void` **actionPerformed**`( java.awt.event.ActionEvent` **a** `)`
- *getDimension*
  `public Dimension` **getDimension**`( )`
- *getID*
  `public int` **getID**`( )`
- *getName*
  `public String` **getName**`( )`
- *getPanel*
  `public JPanel` **getPanel**`( )`
- *getPosition*
  `public Point` **getPosition**`( )`
- *initInfo*
  `public void` **initInfo**`( java.lang.String` **head**`, int` **xlen**`, int` **ylen** `)`
  - **Usage**
    - Initializes a Info canvas (called by constructor).
  - **Parameters**
    - `head` - Name of parameter displayed.
    - `xlen` - Size of canvas in pixels.
    - `ylen` - Size of canvas in pixels.

- *mouseClicked*
  `public void` **mouseClicked**`( java.awt.event.MouseEvent` **me** `)`

- *mouseEntered*
  `public void` **mouseEntered**`( java.awt.event.MouseEvent` **me** `)`

- *mouseExited*
  `public void` **mouseExited**`( java.awt.event.MouseEvent` **me** `)`

- *mousePressed*
  `public void` **mousePressed**`( java.awt.event.MouseEvent` **me** `)`

- *mouseReleased*
  `public void` **mouseReleased**`( java.awt.event.MouseEvent` **me** `)`

- *paintComponent*
  `public void` **paintComponent**`( java.awt.Graphics` **g** `)`
  - **Usage**
    - Called by repaint, calls update.

- *redraw*
  `public void` **redraw**`( )`
  - **Usage**
    - Redraw without changes (repaint).

- *redraw*
  `public void` **redraw**`( int` **severity**`, java.lang.String` **colorname**`, java.lang.String` **value**`, boolean` **draw** `)`
  - **Usage**
    - Redraw with new value
  - **Parameters**
    - `severity` - 0: Display value string only, 1: Header plus value.
    - `colorname` - (Red, Green, Blue, Yellow, Cyan, Magenta).
    - `value` - Short string describing info.
    - `draw` - True: redraw, false: update values only.

- *setActionListener*
  `public void` **setActionListener**`( java.awt.event.ActionListener` **actionlistener** `)`

- *setColor*
  `public void` **setColor**`( java.lang.String` **colorname** `)`
  - **Usage**
    - Set color.
  - **Parameters**
    - `colorname` - (Red, Green, Blue, Yellow, Cyan, Magenta).

- *setColorBack*
  `public void` **setColorBack**`( java.awt.Color` **color** `)`

- *setID*
  `public void` **setID**`( int` **i** `)`
- *setSizeXY*
  `public void` **setSizeXY**`( )`
- *setSizeXY*
  `public void` **setSizeXY**`( java.awt.Dimension` **dd** `)`
- *update*
  `public void` **update**`( java.awt.Graphics` **g** `)`
  - **Usage**
    - Overwriting update method we avoid clearing the graphics. This would cause flickering update is not called by repaint!

1.2.14 Class xInternalCompound

Special internal frame for one to four split panes.

Declaration

public class xInternalCompound
extends xgui.xInternalFrame

Serializable Fields

Constructors

- *xInternalCompound*
  `public` **xInternalCompound**`( java.lang.String` **title**`, javax.swing.ImageIcon` **icon**`, int` **divisions**`, xgui.xLayout` **la**`, java.awt.Color` **back** `)`
  - **Usage**
    - Creates internal frame.
  - **Parameters**
    - `title` - Title.
    - `icon` - Window icon.
    - `divisions` - Controls layout of split panes.
    - `la` - Layout (position and size).
    - `back` - Background color.

METHODS

- *getDividerSize*
  `public int` **getDividerSize**`( )`

- *rebuild*
  `public void` **rebuild**`( javax.swing.JPanel` **panel** `)`
  - **Usage**
    - Replaces panel of internal frame.
  - **Parameters**
    - `panel` -

- *rebuild*
  `public void` **rebuild**`( javax.swing.JPanel` **panel1**`, javax.swing.JPanel` **panel2** `)`
  - **Usage**
    - Creates new container panel, adds the two panels and replaces panel of internal frame.

      Division 0: left 1, right 2

      Division 1: top 1, bottom 2
  - **Parameters**
    - `panel1` -
    - `panel2` -

- *rebuild*
  `public void` **rebuild**`( javax.swing.JPanel` **panel1**`, javax.swing.JPanel` **panel2**`, javax.swing.JPanel` **panel3** `)`
  - **Usage**
    - Creates new container panel, adds the three panels and replaces panel of internal frame.

      Division 0: left 1, right top 2, right bottom 3

      Division 1: top 1, bottom left 2, bottom right 3

      Division 2: top left 1, top right 2, bottom 3
  - **Parameters**
    - `panel1` -
    - `panel2` -
    - `panel3` -

- *rebuild*
  `public void` **rebuild**`( javax.swing.JPanel` **panel1**`, javax.swing.JPanel` **panel2**`, javax.swing.JPanel` **panel3**`, javax.swing.JPanel` **panel4** `)`

  - **Usage**
    * Creates new container panel, adds the four panels and replaces panel of internal frame.

      Only one possibility for arranging four panels.
  - **Parameters**
    * `panel1` -
    * `panel2` -
    * `panel3` -
    * `panel4` -

1.2.15 Class **xInternalFrame**

Frame for one panel. To be added by caller to a JDesktopPane. Provides functions to add one panel.

Declaration

public class xInternalFrame
extends javax.swing.JInternalFrame
implements java.awt.event.ActionListener, javax.swing.event.InternalFrameListener, java.awt.event.ComponentListener

Serializable Fields

Constructors

- *xInternalFrame*
  `public` **xInternalFrame**`( java.lang.String` **title**`, xgui.xLayout` **la** `)`
  - **Usage**
    * Create the frame. Set background and layout.
  - **Parameters**
    * `title` - Title of frame.
    * `la` - Frame layout (position and size).

Methods

- *actionPerformed*
  `public void` **actionPerformed**`( java.awt.event.ActionEvent` **e** `)`
- *addWindow*
  `public void` **addWindow**`( javax.swing.JPanel` **panel** `)`
  - **Usage**

    * Remove all panels, add this one, and pack.
  - **Parameters**
    * `panel` - Panel to display.

- *componentHidden*
  `public void` **componentHidden**`( java.awt.event.ComponentEvent e )`

- *componentMoved*
  `public void` **componentMoved**`( java.awt.event.ComponentEvent e )`

- *componentResized*
  `public void` **componentResized**`( java.awt.event.ComponentEvent e )`
  - **Usage**
    * Set new preferred size when resizable was enabled, otherwise noop.

- *componentShown*
  `public void` **componentShown**`( java.awt.event.ComponentEvent e )`

- *getFrameLayout*
  `public xLayout` **getFrameLayout**`( )`

- *getPanel*
  `public JPanel` **getPanel**`( )`

- *internalFrameActivated*
  `public void` **internalFrameActivated**`( javax.swing.event.InternalFrameEvent e )`

- *internalFrameClosed*
  `public void` **internalFrameClosed**`( javax.swing.event.InternalFrameEvent e )`
  - **Usage**
    * Store layout (position and size)

- *internalFrameClosing*
  `public void` **internalFrameClosing**`( javax.swing.event.InternalFrameEvent e )`
  - **Usage**
    * Store layout (position and size)

- *internalFrameDeactivated*
  `public void` **internalFrameDeactivated**`( javax.swing.event.InternalFrameEvent e )`

- *internalFrameDeiconified*
  `public void` **internalFrameDeiconified**`( javax.swing.event.InternalFrameEvent e )`

- *internalFrameIconified*
  `public void` **internalFrameIconified**`( javax.swing.event.InternalFrameEvent e )`

- *internalFrameOpened*
  `public void` **internalFrameOpened**`( javax.swing.event.InternalFrameEvent e )`

- *setupFrame*
  `public void` **setupFrame**`( javax.swing.ImageIcon` **icon**`, javax.swing.JMenuBar` **menu**`, javax.swing.JPanel` **panel**`, boolean` **resize** `)`
  - **Usage**
    - Set up frame and add panel.
  - **Parameters**
    - `icon` - Give it an icon.
    - `menu` - Optional menu bar. Event handler defined from caller.
    - `panel` - Panel to be displayed.
    - `resize` - Make frame resizable.

1.2.16 Class xLayout

Layout objects keep information about the appearance of panels: Position, size, columns, and visibility. Layouts are managed by xSet. Layout can be stored/retrieved to/from XML file.

Declaration

public class xLayout
extends java.lang.Object

Constructors

- *xLayout*
  `public` **xLayout**`( java.lang.String` **Name** `)`
  - **Usage**
    - Create layout object with a name.

Methods

- *getColumns*
  `public int` **getColumns**`( )`
- *getName*
  `public String` **getName**`( )`
- *getPosition*
  `public Point` **getPosition**`( )`
- *getSize*
  `public Dimension` **getSize**`( )`
- *set*
  `public void` **set**`( java.awt.Point` **lpos**`, java.awt.Dimension` **lsize**`, int` **columns**`, boolean` **lshow** `)`
  - **Usage**

    * Set layout.
  - **Parameters**
    * `lpos` - Position or null.
    * `lsize` - Size or null.
    * `columns` - Columns or 0.
    * `lshow` - visibility.

- *show*
  `public boolean` **show**`( )`

- *toString*
  `public String` **toString**`( )`
  - **Returns** - Formatted string

    name:x=,y=,w=,h=,columns=,show=.

- *XmlLine*
  `public String` **XmlLine**`( )`
  - **Returns** - XML formatted line to be inserted in XML file by caller.

    name shape="x,y,w,h" columns="" show=""

1.2.17 Class **xLogger**

Central print function into logger window.

Declaration

public class xLogger
extends java.lang.Object

Constructors

- *xLogger*
  `public` **xLogger**`( )`
  - **Usage**
    * Creates a State canvas.

Methods

- *print*
  `public static final void` **print**`( int` **severity**`, java.lang.String` **s** `)`

- *setLoggerPanel*
  `public static final void` **setLoggerPanel**`( xgui.xPanelLogger` **p** `)`

1.2.18 Class **xMeter**

Graphic item rate meter. Can be displayed in two fixed sizes.

Declaration

public class xMeter
extends javax.swing.JPanel
implements xiPanelItem, java.awt.event.MouseListener, java.awt.event.ActionListener

Serializable Fields

Fields

- public static final int ARC
  - Arc mode = 0, half circle with indicator.
- public static final int BAR
  - Bar mode = 1, horizontal.
- public static final int TREND
  - Trend histogram = 2.
- public static final int STAT
  - Statistics histogram = 3.
- public static final int XSIZE
  - normal size x
- public static final int YSIZE
  - normal size y
- public static final int XSIZE_LARGE
  - large size x
- public static final int YSIZE_LARGE
  - large size y

Constructors

- *xMeter*
  `public` **xMeter(** `int` **mode,** `java.lang.String` **name,** `double` **min,** `double` **max,** `int` **xlength,** `int` **ylength,** `java.awt.Color` **c)**
  - **Usage**
    - Creates a meter canvas.
  - **Parameters**
    - `mode` - BAR, ARC, TREND, STAT
    - `name` - Name of parameter displayed
    - `min` - Parameter value range
    - `max` - Parameter value range
    - `xlength` - Size of canvas in pixels (XSIZE)
    - `ylength` - Size of canvas in pixels (YSIZE)
    - `c` - Color of markers

Methods

- *actionPerformed*
  `public void` **actionPerformed(** `java.awt.event.ActionEvent` **a)**
  - **Usage**
    - Dispatch actions from right mouse button.

- *getAutoScale*
  `public boolean` **getAutoScale( )**

- *getClone*
  `public xMeter` **getClone(** `java.lang.String` **name)**
  - **Usage**
    - Create a new meter from existing one. When clone shall be attached to parent meter, typically these settings must be applied:

      clone.setSizeXY(new Dimension(XSIZE_LARGE,YSIZE_LARGE));

      clone.hasParent();

      parent.setExternMeter(clone,Frame);
  - **Parameters**
    - `name` - Name of clone.

- *getColor*
  `public String` **getColor( )**

- *getDimension*
  `public Dimension` **getDimension( )**

- *getHead1*
  `public String` **getHead1( )**

- *getHead2*
  `public String` **getHead2**`( )`
- *getHead3*
  `public String` **getHead3**`( )`
- *getID*
  `public int` **getID**`( )`
- *getLogScale*
  `public boolean` **getLogScale**`( )`
- *getMax*
  `public double` **getMax**`( )`
- *getMin*
  `public double` **getMin**`( )`
- *getMode*
  `public int` **getMode**`( )`
- *getName*
  `public String` **getName**`( )`
- *getPanel*
  `public JPanel` **getPanel**`( )`
- *getPosition*
  `public Point` **getPosition**`( )`
- *hasParent*
  `protected void` **hasParent**`( )`
  - **Usage**
    * If caller makes a clone of this Meter (parent), and wants that clone to be controlled by this Meter (updated), it must set the clone to hasParent. The clone is passed to its parent by setExternMeter together with a reference to a JInternalFrame where the clone is displayed. This frame must be created by caller. If frame is closed, the parent removes its child.
- *mouseClicked*
  `public void` **mouseClicked**`( java.awt.event.MouseEvent` **me** `)`
  - **Usage**
    * Not used.
- *mouseEntered*
  `public void` **mouseEntered**`( java.awt.event.MouseEvent` **me** `)`
  - **Usage**
    * Not used.
- *mouseExited*
  `public void` **mouseExited**`( java.awt.event.MouseEvent` **me** `)`
  - **Usage**

    * Not used.

- *mousePressed*
  `public void` **mousePressed**`( java.awt.event.MouseEvent` **me** `)`
  - **Usage**
    * Pull down context menu with right mouse button

- *mouseReleased*
  `public void` **mouseReleased**`( java.awt.event.MouseEvent` **me** `)`
  - **Usage**
    * Not used.

- *paintComponent*
  `public void` **paintComponent**`( java.awt.Graphics` **g** `)`
  - **Usage**
    * Called by repaint, calls update.

- *redraw*
  `public void` **redraw**`( )`
  - **Usage**
    * Redraw without changes.

- *redraw*
  `public void` **redraw**`( double` **value** `)`
  - **Usage**
    * Final redraw with new value. Called by all other redraw entries.
  - **Parameters**
    * `value` - New value.

- *redraw*
  `public void` **redraw**`( double` **value**`, boolean` **valid**`, boolean` **draw** `)`
  - **Usage**
    * Redraw with new value.
  - **Parameters**
    * `value` - New value
    * `valid` - True if value is valid, false if not. Change color on change in validity.
    * `draw` - True: call redraw, otherwise no repaint.

- *setActionListener*
  `public void` **setActionListener**`( java.awt.event.ActionListener` **actionlistener** `)`

- *setAutoScale*
  `public void` **setAutoScale**`( boolean` **as** `)`

- *setColor*
  `public void` **setColor**`( java.awt.Color` **color** `)`

  - **Usage**
    * Set color of marker.
  - **Parameters**
    * `color` -

- *setColor*
`public void` **setColor**`( java.lang.String` **colorname** `)`

  - **Usage**
    * Set color by name.
  - **Parameters**
    * `colorname` - (Red, Green, Blue, Yellow, Cyan, Magenta).

- *setColorBack*
`public void` **setColorBack**`( java.awt.Color` **color** `)`

  - **Usage**
    * Set back ground color. Color for text is two times brighter. Background color should therefore be dark.
  - **Parameters**
    * `color` -

- *setDefaultAutoScale*
`public void` **setDefaultAutoScale**`( boolean` **as** `)`

- *setDefaultLimits*
`public void` **setDefaultLimits**`( double` **min**`, double` **max** `)`

  - **Usage**
    * Initializes a meter canvas with new limits and store as default.
  - **Parameters**
    * `min` - parameter value range
    * `max` - parameter value range

- *setDefaultLogScale*
`public void` **setDefaultLogScale**`( boolean` **log** `)`

- *setDefaultMode*
`public void` **setDefaultMode**`( int` **mode** `)`

  - **Usage**
    * Set mode of presentation.
  - **Parameters**
    * `mode` - ARC, BAR, TREND, STAT

- *setDefaults*
`public void` **setDefaults**`( )`

  - **Usage**
    * Resets log scale, limits, autoscale, and mode to values as set by default settings.

- *setExternMeter*
  `protected void` **setExternMeter**`( xgui.xMeter` **extmeter**`, javax.swing.JInternalFrame` **extframe** `)`
  - **Usage**
    - Attach a second meter to this. Second meter will be updated from this.
  - **Parameters**
    - `extmeter` - Typically a clone with large size to be displayed in an extra frame.
    - `extframe` - Extra frame for the clone. Must be handled by caller.

- *setID*
  `public void` **setID**`( int` **i** `)`

- *setInterval*
  `public void` **setInterval**`( int` **seconds** `)`
  - **Usage**
    - Set update interval for trending and histogramming
  - **Parameters**
    - `seconds` - Time interval.

- *setLettering*
  `public void` **setLettering**`( java.lang.String` **node**`, java.lang.String` **appl**`, java.lang.String` **name**`, java.lang.String` **units** `)`
  - **Usage**
    - Set some strings for lettering. In the compact mode Node:Appl and Name are shown in two lines. Units are not shown. In large mode Node, Appl and Name are shown in three lines, units are shown.
  - **Parameters**
    - `node` - Node name. Get back by getHead1().
    - `appl` - Application name. Get back by getHead2().
    - `name` - Parameter name. Get back by getHead3().
    - `units` - Units

- *setLimits*
  `public void` **setLimits**`( double` **min**`, double` **max** `)`
  - **Usage**
    - Initializes a meter canvas with new limits.
  - **Parameters**
    - `min` - parameter value range
    - `max` - parameter value range

- *setLogScale*
  `public void` **setLogScale**`( boolean` **log** `)`

- *setMode*
  `public void` **setMode**`( int` **mode** `)`
  - **Usage**
    - Set mode of presentation.

  - **Parameters**
    * `mode` - ARC, BAR, TREND, STAT

- *setName*
  `public void` **setName**`( java.lang.String` **sH** `)`

- *setSizeXY*
  `public void` **setSizeXY**`( )`

- *setSizeXY*
  `public void` **setSizeXY**`( java.awt.Dimension` **dd** `)`

- *setUnits*
  `public void` **setUnits**`( java.lang.String` **units** `)`
  - **Usage**
    * Set units string.
  - **Parameters**
    * `units` -

- *update*
  `public void` **update**`( java.awt.Graphics` **g** `)`
  - **Usage**
    * Overwriting update method we avoid clearing the graphics. This would cause flickering. Update is not called by repaint!

1.2.19 Class xPanelCommand

Panel for command tree

Declaration

public class xPanelCommand **extends** javax.swing.JPanel **implements** javax.swing.event.TreeSelectionListener, java.awt.event.ActionListener

Serializable Fields

Constructors

- *xPanelCommand*
  `public` **xPanelCommand**`( xgui.xDimBrowser` **browser**`, java.awt.Dimension` **dim** `)`
  - **Usage**

    * Opens panel for command tree. Gets list of xDimCommand objects
  - **Parameters**
    * `browser` - DIM browser.
    * `dim` - Size and position of window.

METHODS

- *actionPerformed*
  `public void` **actionPerformed**`( java.awt.event.ActionEvent  e )`
  - **Usage**
    * Handles mainly RET to execute commands.

- *setCommandDescriptors*
  `protected void` **setCommandDescriptors**`( java.util.Vector  desc )`
  - **Usage**
    * Called by desktop, descriptors from PanelParameter
  - **Parameters**
    * `desc` - Descriptor list as returned from xPanelParameter.getCommandDescriptors()
  - **See Also**
    * `xgui.xPanelParameter` (in 1.2.28, page 67)

- *setUserCommand*
  `protected void` **setUserCommand**`( xgui.xiUserCommand  format )`
  - **Usage**
    * Called by desktop, format from xiUserPanel.getUserCommand()
  - **Parameters**
    * `format` - format.getArgumentStyleXml(...) function is called before command argument composing to check if arguments should be formatted in XML or not.
  - **See Also**
    * `xgui.xiUserPanel` (in 1.1.10, page 15)

- *valueChanged*
  `public void` **valueChanged**`( javax.swing.event.TreeSelectionEvent  e )`

1.2.20 CLASS xPanelDabc

Form panel to control DABC.

DECLARATION

public class xPanelDabc **extends** xgui.xPanelPrompt **implements** java.awt.event.ActionListener, java.lang.Runnable

Serializable Fields

Constructors

- *xPanelDabc*
  `public` **xPanelDabc**`( java.lang.String` **title**`, xgui.xDimBrowser` **diminfo**`, xgui.xiDesktop` **desktop**`, java.awt.event.ActionListener` **al** `)`
  - **Usage**
    - Constructor of DABC launch panel.
  - **Parameters**
    - `title` - Title of window.
    - `diminfo` - DIM browser
    - `desktop` - Interface to desktop
    - `al` - Event handler of desktop. Handles events from xTimer.

      Passed actions are: Update, DisplayFrame, RemoveFrame.
  - **See Also**
    - `xgui.xTimer` (in 1.2.46, page 112)

Methods

- *actionPerformed*
  `public void` **actionPerformed**`( java.awt.event.ActionEvent` **e** `)`
  - **Usage**
    - Handle events.
  - **Parameters**
    - `e` - Event. Some events are handled directly. Others are handled in a thread. If an update of DIM parameter list is necessary, Update event is launched through timer and handled by desktop action listener.

      "ReadSetup":

      Creates a new xSetup object, read Xdaq XML setup file, get references to name/type/value lists. Create for each context a xPanelSetup passing the list references, and display in a separate frame.

      "DabcSave":

      Save content of form to file and contents of context panels to XML file.

- *releaseDimServices*
  `public void` **releaseDimServices**`( )`
  - **Usage**

* Called in xDesktop to release references to DIM services.

- *run*
  `public void` **`run( )`**
  - **Usage**
    * Thread handling events.

      If an update of DIM parameter list is necessary, Update event is launched through timer and handled by desktop action listener.

- *setDimServices*
  `public void` **`setDimServices( )`**
  - **Usage**
    * Called in xDesktop to rebuild references to DIM services.

1.2.21 Class **xPanelDabcMbs**

Form panel to control DABC.

Declaration

public class xPanelDabcMbs
extends xgui.xPanelPrompt
implements java.awt.event.ActionListener, java.lang.Runnable

Serializable Fields

Constructors

- *xPanelDabcMbs*
  `public` **`xPanelDabcMbs(`** `java.lang.String` **`title,`** `xgui.xDimBrowser` **`diminfo,`** `xgui.xiDesktop` **`desktop,`** `java.awt.event.ActionListener` **`al )`**
  - **Usage**
    * Constructor of MBS+DABC launch panel.
  - **Parameters**
    * `title` - Title of window.
    * `diminfo` - DIM browser
    * `desktop` - Interface to desktop
    * `al` - Event handler of desktop. Handles events from xTimer.

      Passed actions are: Update, DisplayFrame, RemoveFrame.
  - **See Also**
    * `xgui.xTimer` (in 1.2.46, page 112)

Methods

- *actionPerformed*
  `public void` **actionPerformed**`( java.awt.event.ActionEvent  e )`
  - **Usage**
    - Handle events.
  - **Parameters**
    - `e` - Event. Some events are handled directly. Others are handled in a thread. If an update of DIM parameter list is necessary, Update event is launched through timer and handled by desktop action listener.

- *releaseDimServices*
  `public void` **releaseDimServices**`( )`
  - **Usage**
    - Called in xDesktop to release references to DIM services.

- *run*
  `public void` **run**`( )`

- *setDimServices*
  `public void` **setDimServices**`( )`
  - **Usage**
    - Called in xDesktop to rebuild references to DIM services.

1.2.22 Class xPanelGraphics

Container panel for graphic panels. Provides functions to add graphic panels in columns.

Declaration

public class xPanelGraphics
extends javax.swing.JPanel
implements java.awt.event.ActionListener

Serializable Fields

Constructors

- *xPanelGraphics*
  `public` **xPanelGraphics**`( java.awt.Dimension  dim, int  col )`
  - **Usage**

    * Create panel for number of columns.
  - **Parameters**
    * `dim` - Dimension
    * `col` - Columns

METHODS

- *actionPerformed*
  `public void` **actionPerformed**`( java.awt.event.ActionEvent` **e** `)`

- *addGraphics*
  `public void` **addGraphics**`( xgui.xiPanelItem` **panelItem**`, boolean` **update** `)`
  - **Usage**
    * Add graphic panel to list.
  - **Parameters**
    * `panelItem` - Interface of panel to be added. setSizeXY and setID is called.
    * `update` - True: update all, false to only add to list.

- *cleanup*
  `public void` **cleanup**`( )`
  - **Usage**
    * Remove all panels from list and panel.

- *createMenuBar*
  `public JMenuBar` **createMenuBar**`( )`

- *getColumns*
  `public int` **getColumns**`( )`

- *removeGraphics*
  `public void` **removeGraphics**`( javax.swing.JPanel` **panel** `)`
  - **Usage**
    * Remove a panel from list and update all.
  - **Parameters**
    * `panel` - Panel to be removed.

- *setColorBack*
  `public void` **setColorBack**`( java.awt.Color` **back** `)`

- *setListener*
  `public void` **setListener**`( java.awt.event.ActionListener` **al** `)`
  - **Usage**
    * Attach an action listener. This is called directly after creating the internal frame where this is in. Listener is internal frame.
  - **Parameters**
    * `al` - Action listener.
  - **See Also**

        * `xgui.xInternalFrame` (in 1.2.15, page 44)

- *updateAll*
  `public void` **updateAll**`( )`
  - **Usage**
    * Remove all, adjust size, add to panel, revalidate, call action handler (null).

1.2.23 Class xPanelHisto

Panel for set of histogram panels.

Declaration

public class xPanelHisto **extends** javax.swing.JPanel **implements** java.awt.event.ActionListener

Serializable Fields

Constructors

- *xPanelHisto*
  `public` **xPanelHisto**`( java.awt.Dimension` **dim**`, int` **col** `)`
  - **Usage**
    * Uses JPanel with GridBagLayout. Assumes that all items have same size. Items are ordered in lines, each line has same number of columns (except last one). Number of columns define number of lines.
  - **Parameters**
    * `dim` - Dimension
    * `col` - Number of columns

Methods

- *actionPerformed*
  `public void` **actionPerformed**`( java.awt.event.ActionEvent` **e** `)`
  - **Usage**
    * Event handler.

      Large: Called through xHisto handler which wants to be displayed in an extra frame. Histogram is cloned and a new frame is created. This frame will be action handler for that histogram.

- *addHistogram*
  `public void` **addHistogram**`( xgui.xHisto` **histo** `)`
  - **Usage**
    - Add histogram item to internal table. No redraw.
  - **Parameters**
    - `histo` - Histogram.

- *addHistogram*
  `public void` **addHistogram**`( xgui.xHisto` **histo**`, boolean` **update** `)`
  - **Usage**
    - Add histogram item to internal table.
  - **Parameters**
    - `histo` - Histogram.
    - `update` - True: updateAll, false: no graphics update. Several histograms can be added without redrawing the panel each time. The last one should.

- *cleanup*
  `public void` **cleanup**`( )`
  - **Usage**
    - Cleanup item list and panel.

- *createMenuBar*
  `public JMenuBar` **createMenuBar**`( )`

- *getColumns*
  `public int` **getColumns**`( )`

- *removeHistogram*
  `public void` **removeHistogram**`( xgui.xHisto` **histo** `)`
  - **Parameters**
    - `histo` - Removes histogram. Calls updateAll().

- *setColumns*
  `public void` **setColumns**`( int` **col** `)`
  - **Parameters**
    - `col` - New number of columns. Calls updateAll().

- *setListener*
  `public void` **setListener**`( java.awt.event.ActionListener` **actionlistener** `)`
  - **Usage**
    - This is called directly after creating the internal frame where this is in. Listener is xInternalFrame. After resizing the items by "Size" event this action listener is called with null event to pack the frame.
  - **Parameters**
    - `actionlistener` - Frame containing this panel.

- *updateAll*
  `public void` **updateAll**`( )`
  - **Usage**
    - Removes all items from panel, resize and rebuild all items.

1.2.24 Class xPanelInfo

Panel for set of info panels.

Declaration

public class xPanelInfo **extends** javax.swing.JPanel **implements** java.awt.event.ActionListener

Serializable Fields

Constructors

- *xPanelInfo*
  `public` **xPanelInfo**`( java.awt.Dimension` **dim** `)`
  - **Usage**
    - Uses JPanel with GridBagLayout. Assumes that all items have same size. One item per line.
  - **Parameters**
    - `dim` - Dimension

Methods

- *actionPerformed*
  `public void` **actionPerformed**`( java.awt.event.ActionEvent` **e** `)`
- *addInfo*
  `public void` **addInfo**`( xgui.xInfo` **info** `)`
  - **Usage**
    - Add info item to internal table, calls setSizeXY and setID.
  - **Parameters**
    - `info` - Info. .
- *addInfo*
  `public void` **addInfo**`( xgui.xInfo` **info**`, boolean` **update** `)`

  - **Usage**
    * Add info item to internal table.
  - **Parameters**
    * `info` - Info.
    * `update` - True: updateAll, false: no graphics update. Several Infos can be added without redrawing the panel each time. The last one should.

- *cleanup*
  `public void` **cleanup**`( )`

  - **Usage**
    * Cleanup item list and panel.

- *createMenuBar*
  `public JMenuBar` **createMenuBar**`( )`

- *removeInfo*
  `public void` **removeInfo**`( xgui.xInfo` **info** `)`

  - **Parameters**
    * `info` - Removes Info. Calls updateAll().

- *setListener*
  `public void` **setListener**`( java.awt.event.ActionListener` **al** `)`

- *updateAll*
  `public void` **updateAll**`( )`

  - **Usage**
    * Removes all items from panel, resize and rebuild all items.

1.2.25 Class **xPanelLogger**

DIM GUI class

Declaration

public class xPanelLogger
extends javax.swing.JPanel
implements java.awt.event.ActionListener

Serializable Fields

CONSTRUCTORS

- *xPanelLogger*
  `public` **xPanelLogger**`( java.awt.Dimension` **dim** `)`
  - **Usage**
    * DIM GUI class. Uses JScrollPanes with GridBagLayout. Creates the table for the parameters and the tree for the commands. Creates rate meters for dataBW, dataLatency and dataRate

METHODS

- *actionPerformed*
  `public void` **actionPerformed**`( java.awt.event.ActionEvent` **e** `)`
- *createMenuBar*
  `public JMenuBar` **createMenuBar**`( )`
- *internalFrameClosed*
  `public void` **internalFrameClosed**`( javax.swing.event.InternalFrameEvent` **e** `)`
- *print*
  `public void` **print**`( java.lang.String` **s** `)`
- *setListener*
  `public void` **setListener**`( java.awt.event.ActionListener` **al** `)`

1.2.26 CLASS xPanelMbs

Form panel to control MBS.

DECLARATION

public class xPanelMbs
extends xgui.xPanelPrompt
implements java.awt.event.ActionListener, java.lang.Runnable

SERIALIZABLE FIELDS

CONSTRUCTORS

- *xPanelMbs*
  `public` **xPanelMbs**`( java.lang.String` **title**`, xgui.xDimBrowser` **diminfo**`, xgui.xiDesktop` **desktop**`, java.awt.event.ActionListener` **al** `)`

- **Usage**
  - Constructor of MBS launch panel.
- **Parameters**
  - `title` - Title of window.
  - `diminfo` - DIM browser
  - `desktop` - Interface to desktop
  - `al` - Event handler of desktop. Handles events from xTimer.

    Passed actions are: Update, DisplayFrame, RemoveFrame.

Methods

- *actionPerformed*
  `public void` **actionPerformed**`( java.awt.event.ActionEvent  e )`
  - **Usage**
    - Handle events.
  - **Parameters**
    - `e` - Event. Some events are handled directly. Others are handled in a thread. If an update of DIM parameter list is necessary, Update event is launched through timer and handled by desktop action listener.

- *releaseDimServices*
  `public void` **releaseDimServices**`( )`
  - **Usage**
    - Called in xDesktop to release references to DIM services.

- *run*
  `public void` **run**`( )`
  - **Usage**
    - Thread handling events.

      If an update of DIM parameter list is necessary, Update event is launched through timer and handled by desktop action listener.

- *setDimServices*
  `public void` **setDimServices**`( )`
  - **Usage**
    - Called in xDesktop to rebuild references to DIM services.

1.2.27 Class xPanelMeter

Panel for set of meter panels.

Declaration

public class xPanelMeter
extends javax.swing.JPanel
implements java.awt.event.ActionListener

Serializable Fields

Constructors

- *xPanelMeter*
  `public` **xPanelMeter**`( java.awt.Dimension` **dim**`, int` **col** `)`
  - **Usage**
    - Uses JPanel with GridBagLayout. Assumes that all items have same size. Items are ordered in lines, each line has same number of columns (except last one). Number of columns define number of lines.
  - **Parameters**
    - `dim` - Dimension
    - `col` - Number of columns

Methods

- *actionPerformed*
  `public void` **actionPerformed**`( java.awt.event.ActionEvent` **e** `)`
  - **Usage**
    - React to menu selections. If this is set as action listener of the xMeters, it processes event "Large" from the xMeter.

      Event "Zoom" is fired by the menu bar.

- *addMeter*
  `public void` **addMeter**`( xgui.xMeter` **meter** `)`
  - **Usage**
    - Add meter item to internal table. No redraw, calls setSizeXY and setID.
  - **Parameters**
    - `meter` - Meter.

- *addMeter*
  `public void` **addMeter**`( xgui.xMeter` **meter**`, boolean` **update** `)`
  - **Usage**
    - Add meter item to internal table.

  - **Parameters**
    * `meter` - Meter.
    * `update` - True: updateAll, false: no graphics update. Several meters can be added without redrawing the panel each time. The last one should.

- *cleanup*
  `public void` **cleanup**`( )`

  - **Usage**
    * Cleanup item list and panel.

- *createMenuBar*
  `public JMenuBar` **createMenuBar**`( )`

- *getColumns*
  `public int` **getColumns**`( )`

- *removeMeter*
  `public void` **removeMeter**`( xgui.xMeter` **meter** `)`

  - **Parameters**
    * `meter` - Removes meter. Calls updateAll().

- *setColumns*
  `public void` **setColumns**`( int` **col** `)`

  - **Parameters**
    * `col` - New number of columns. Calls updateAll().

- *setListener*
  `public void` **setListener**`( java.awt.event.ActionListener` **actionlistener** `)`

  - **Usage**
    * This is called directly after creating the internal frame where this is in. Listener is xInternalFrame. After resizing the items by "Zoom" event this action listener is called with same event to pack the frame.
  - **Parameters**
    * `actionlistener` - Frame containing this panel.

- *updateAll*
  `public void` **updateAll**`( )`

  - **Usage**
    * Removes all items from panel, resize and rebuild all items.

1.2.28 Class xPanelParameter

Handles parameter table. On parameter update, the table is updated.

Declaration

public class xPanelParameter
extends javax.swing.JPanel
implements javax.swing.event.TableModelListener

Serializable Fields

Constructors

- *xPanelParameter*
  `public` **xPanelParameter**`( xgui.xDimBrowser` **browser**`, java.awt.Dimension` **dim** `)`
  - **Usage**
    - Panel for parameter table. Gets xDimParameter list from browser and calls initPanel. Polls over all parameters until all have been updated at least once (quality valid). Cleans all graphics panels and build new table. Graphics panels must be rebuilt from caller. Keeps a list of command descriptors (xXmlParser).
  - **Parameters**
    - `browser` - DIM browser.
    - `dim` - Size and position of window.

Methods

- *getCommandDescriptors*
  `public Vector` **getCommandDescriptors**`( )`
  - **Usage**
    - Called from desktop, the list is passed to xPanelCommand.
  - **Returns** - List of command descriptors (xXmlParser)

- *saveColWidth*
  `public void` **saveColWidth**`( )`
  - **Usage**
    - Called by desktop before saving layout. stores column widths in xSet

---

- *tableChanged*
  `public void` **tableChanged**`( javax.swing.event.TableModelEvent` **e** `)`
  - **Usage**
    - If the parameter is not changable, value is ”-” Therefore we must ignore value ”-”. Effect is that a value of ”-” cannot be set to a parameter.

1.2.29 Class **xPanelPrompt**

Base class for prompt panels. Provides functions to build forms from text input fields (one column) and buttons (tool bar).

Declaration

public class xPanelPrompt
extends javax.swing.JPanel

Serializable Fields

Constructors

- *xPanelPrompt*
  `public` **xPanelPrompt**`( java.lang.String` **title** `)`
  - **Usage**
    - Create panel with title.
  - **Parameters**
    - `title` - Title

- *xPanelPrompt*
  `public` **xPanelPrompt**`( java.lang.String` **title**`, java.awt.Dimension` **dim** `)`
  - **Usage**
    - Create panel with title.
  - **Parameters**
    - `title` - Title
    - `dim` - Window size.

Methods

- *addButton*
  `public void` **addButton**`( java.lang.String` **cmd**`, java.lang.String` **tt**`, javax.swing.ImageIcon` **icon**`, java.awt.event.ActionListener` **al** `)`
  - **Usage**
    - Add tool bar button.
  - **Parameters**
    - `cmd` - Button command.
    - `icon` - Button icon.
    - `tt` - Tool tip text.
    - `al` - Action listener handling button command.

- *addCheckBox*
  `public void` **addCheckBox**`( java.lang.String` **label**`, javax.swing.JCheckBox` **check** `)`
  - **Usage**
    - Add check box in column.
  - **Parameters**
    - `label` - Label for check box.
    - `check` - Check box. Action listener must be set from caller.

- *addCheckBox*
  `public JCheckBox` **addCheckBox**`( java.lang.String` **label**`, java.lang.String` **cmd**`, java.awt.event.ActionListener` **al** `)`
  - **Usage**
    - Add check box in column and return it.
  - **Parameters**
    - `label` - Label for check box.
    - `cmd` - Action command.
    - `al` - Action listener to handle check box command.
  - **Returns** - Check box.

- *addPrompt*
  `public void` **addPrompt**`( java.lang.String` **label**`, javax.swing.JTextField` **input** `)`
  - **Usage**
    - Add prompter line in column.
  - **Parameters**
    - `label` - Label for prompt.
    - `input` - Input text field. Action listener must be set from caller.

- *addPrompt*
  `public JTextField` **addPrompt**`( java.lang.String` **label**`, java.lang.String` **value**`, java.lang.String` **cmd**`, int` **width**`, java.awt.event.ActionListener` **al** `)`
  - **Usage**
    - Add prompter line in column and return text field.
  - **Parameters**
    - `label` - Label for prompt.
    - `value` - Default value.
    - `cmd` - Action command.
    - `width` - Size of text field.
    - `al` - Action listener to handle action command.
  - **Returns** - Text field.

- *addTextButton*
  `public void` **addTextButton**`( java.lang.String` **label**`, java.lang.String` **cmd**`, java.lang.String` **tt**`, java.awt.event.ActionListener` **al** `)`

  - **Usage**
    * Add text button in column.
  - **Parameters**
    * `label` - Text of button.
    * `cmd` - Button command.
    * `tt` - Tool tip text.
    * `al` - Action listener handling button command.

- *askInput*
  `public String` **askInput**`( java.lang.String` **title** `)`

  - **Usage**
    * Modal pop up window to enter text.
  - **Parameters**
    * `title` - Text describing input.

- *askInput*
  `public String` **askInput**`( java.lang.String` **title**`, java.lang.String` **Default** `)`

  - **Usage**
    * Modal pop up window to enter text.
  - **Parameters**
    * `title` - Text describing input.
    * `Default` - Default text.

- *askQuestion*
  `public boolean` **askQuestion**`( java.lang.String` **title**`, java.lang.String` **question** `)`

  - **Usage**
    * Modal pop up window with question.
  - **Parameters**
    * `title` - Text.
    * `question` - Question.
  - **Returns** - True, if answer was yes.

- *setTitle*
  `public void` **setTitle**`( java.lang.String` **title**`, java.awt.Color` **color** `)`

  - **Usage**
    * Set title and color of title boarder.
  - **Parameters**
    * `title` - Title
    * `color` - Color

- *tellError*
  `public void` **tellError**`( java.lang.String` **msg** `)`

  - **Usage**
    * Modal pop up window with error message.
  - **Parameters**

        * `msg` - Error message

- *tellInfo*
  `public void` **tellInfo**`( java.lang.String` **msg** `)`
  - **Usage**
    * Modal pop up window with info message.
  - **Parameters**
    * `msg` - Info message

1.2.30 Class xPanelSelect

Panel for display of selected list of parameters.

Declaration

public class xPanelSelect **extends** xgui.xPanelPrompt **implements** xiUserInfoHandler, java.awt.event.ActionListener

Serializable Fields

Constructors

- *xPanelSelect*
  `public` **xPanelSelect**`( java.lang.String` **file**`, java.lang.String` **title**`,`
  `xgui.xiDimBrowser` **browser**`, xgui.xiDesktop` **desktop**`,`
  `java.awt.event.ActionListener` **actionlistener** `)`
  - **Usage**
    * Creates panel to specify filters for parameters and display a filtered parameter list.
  - **Parameters**
    * `file` - XML file to restore values (Selection.xml).
    * `title` - Title of panel.
    * `browser` - DIM browser interface.
    * `desktop` - Desktop interface to open the windows.
    * `actionlistener` - Events can be passed to Desktop action listener.

Methods

- *actionPerformed*
  `public void` **actionPerformed**`( java.awt.event.ActionEvent` **e** `)`
- *infoHandler*
  `public void` **infoHandler**`( xgui.xiDimParameter` **param** `)`

  - **Usage**
    * Call back for parameter update (xiUserInfoHandler).

- *init*
`public void` **init**`( xgui.xiDesktop` **desktop**`, java.awt.event.ActionListener` **actionlistener** `)`
  - **Usage**
    * Called by constructor.
  - **Parameters**
    * `desktop` - Desktop interface to open the windows.
    * `actionlistener` - Events can be passed to Desktop action listener.

- *releaseDimServices*
`public void` **releaseDimServices**`( )`
  - **Usage**
    * Release local references to DIM parameters and commands (xiUserPanel) otherwise we would get memory leaks!

- *restoreSetup*
`public void` **restoreSetup**`( java.lang.String` **file** `)`
  - **Usage**
    * Reads Xml file and restores filter setup values.
  - **Parameters**
    * `file` - Xml file name.

- *saveSetup*
`public void` **saveSetup**`( java.lang.String` **file** `)`
  - **Usage**
    * Writes Xml file with filter setup values.
  - **Parameters**
    * `file` - Xml file name.

- *setDimServices*
`public void` **setDimServices**`( )`
  - **Usage**
    * Setup references to DIM parameters and commands (xiUserPanel) Called after releaseDimServices() after every change in DIM services

1.2.31 Class **xPanelSetup**

Panel to display context from Xdaq XML file as editable textfields.

Declaration

public class xPanelSetup
extends xgui.xPanelPrompt
implements java.awt.event.ActionListener

Serializable Fields

Constructors

- *xPanelSetup*
  `protected` **xPanelSetup**`( java.lang.String` **title**`, java.util.Vector` **names**`, java.util.Vector` **types**`, java.util.Vector` **values**`, int` **offset**`, int` **number** `)`
  - **Usage**
    - Create panel for one context. In the lists there are all contexts. Therefore an offset and length must be passed.
  - **Parameters**
    - `title` - Title of frame.
    - `names` - Reference to name list.
    - `types` - Reference to type list.
    - `values` - Reference to value list.
    - `offset` - Offset in the lists.
    - `number` - Items in the lists to be used (behind offset).
  - **See Also**
    - `xgui.xPanelDabc` (in 1.2.20, page 55)

Methods

- *actionPerformed*
  `public void` **actionPerformed**`( java.awt.event.ActionEvent` **e** `)`
- *updateList*
  `protected void` **updateList**`( )`
  - **Usage**
    - Update value list from text input fields.
  - **See Also**
    - `xgui.xPanelDabc` (in 1.2.20, page 55)

1.2.32 Class xPanelState

Container panel for State panels. Provides functions to add graphic panels in ciolumns.

Declaration

public class xPanelState
extends javax.swing.JPanel
implements java.awt.event.ActionListener

Serializable Fields

Constructors

- *xPanelState*
  `public` **xPanelState**`( java.awt.Dimension` **dim**`, int` **col** `)`
  - **Usage**
    - Create panel for number of columns.
  - **Parameters**
    - `dim` - Dimension
    - `col` - Columns

Methods

- *actionPerformed*
  `public void` **actionPerformed**`( java.awt.event.ActionEvent` **e** `)`
  - **Usage**
    - React to menu selections. Event "Fit" fired by the menu bar calls event handler of associated frame to pack.

      Event "Text" fired by the menu bar switches text display on/off.

      Others change number of columns.
  - **See Also**
    - `xgui.xInternalFrame` (in 1.2.15, page 44)

- *addState*
  `public void` **addState**`( xgui.xState` **state** `)`
  - **Usage**
    - Add graphic panel to list.
  - **Parameters**
    - `state` - State panel to be added. setSizeXY and setID is called.

- *addState*
  `public void` **addState**`( xgui.xState` **state**`, boolean` **update** `)`

  - **Usage**
    * Add graphic panel to list.
  - **Parameters**
    * `state` - State panel to be added. setSizeXY and setID is called.
    * `update` - True: update all, false to only add to list.

- *cleanup*
  `public void` **cleanup**`( )`
  - **Usage**
    * Remove all panels from list and panel.

- *createMenuBar*
  `public JMenuBar` **createMenuBar**`( )`

- *getColumns*
  `public int` **getColumns**`( )`

- *removeState*
  `public void` **removeState**`( xgui.xState` **state** `)`
  - **Usage**
    * Remove a panel from list and update all.
  - **Parameters**
    * `state` - Panel to be removed.

- *setColumns*
  `public void` **setColumns**`( int` **col** `)`

- *setListener*
  `public void` **setListener**`( java.awt.event.ActionListener` **actionlistener** `)`
  - **Usage**
    * This is called directly after creating the internal frame where this is in. Listener is xInternalFrame. After resizing the items this action listener is called pack the frame.
  - **Parameters**
    * `actionlistener` - Frame containing this panel.
  - **See Also**
    * `xgui.xInternalFrame` (in 1.2.15, page 44)

- *updateAll*
  `public void` **updateAll**`( )`
  - **Usage**
    * Redraw all, remove all, adjust size, add to panel, revalidate, call action handler (null).

1.2.33 Class xParser

Parser for (de)composing DIM service names. The DIM service names of DABC follow a defined structure, mainly Dns/Node[:NodeId]/[applNS::]Application[:ApplicationId]/Name.

Declaration

public class xParser
extends java.lang.Object
implements xiParser

Fields

- public static final boolean PARSE_STORE_FULL
  - Parser steering switch to store full name inside.
- public static final boolean PARSE_ONLY
  - Parser steering switch to parse only.
- public static final boolean IS_COMMAND
  - Used for command style names.
- public static final boolean IS_PARAMETER
  - Used for parameter style names.
- public static final boolean MAKE_FULL
  - Build name from components.
- public static final boolean COPY_FULL
  - Do not build name from components
- public static final int NOTSPEC
  - Values for DIM quality mask 0xff (state field): state undefined.
- public static final int SUCCESS
  - Values for DIM quality mask 0xff (state field): success.
- public static final int INFORMATION
  - Values for DIM quality mask 0xff (state field): info.
- public static final int WARNING
  - Values for DIM quality mask 0xff (state field): warning.
- public static final int ERROR
  - Values for DIM quality mask 0xff (state field): error.
- public static final int FATAL
  - Values for DIM quality mask 0xff (state field): fatal.
- public static final int ATOMIC
  - Values for DIM quality mask 0xff00 (type field): atomic data type.
- public static final int GENERIC

  - Values for DIM quality mask 0xff00 (type field): generic data encoded in string (XML).
- public static final int STATE
  - Values for DIM quality mask 0xff00 (type field): state structure.
- public static final int RATE
  - Values for DIM quality mask 0xff00 (type field): rate structure.
- public static final int HISTOGRAM
  - Values for DIM quality mask 0xff00 (type field): histogram structure.
- public static final int MODULE
  - Values for DIM quality mask 0xff00 (type field): module structure.
- public static final int PORT
  - Values for DIM quality mask 0xff00 (type field): port structure.
- public static final int DEVICE
  - Values for DIM quality mask 0xff00 (type field): device structure.
- public static final int QUEUE
  - Values for DIM quality mask 0xff00 (type field): queue structure.
- public static final int COMMANDDESC
  - Values for DIM quality mask 0xff00 (type field): command descriptor.
- public static final int INFO
  - Values for DIM quality mask 0xff00 (type field): info structure.
- public static final int VISIBLE
  - Values for DIM quality mask 0xff0000 (visibility field): to show.
- public static final int MONITOR
  - Values for DIM quality mask 0xff0000 (visibility field): to monitor.
- public static final int CHANGABLE
  - Values for DIM quality mask 0xff0000 (visibility field): can be modified.
- public static final int IMPORTANT
  - Values for DIM quality mask 0xff0000 (visibility field): is important.
- public static final int LOGGING
  - Values for DIM quality mask 0xff0000 (visibility field): changes in logfile.
- public static final int NOMODE
  - Values for DIM quality mask 0xff000000 (mode field): not used.

Constructors

- *xParser*
  `public` **xParser**`( )`
  - **Usage**
    - Create an empty parser object.

- *xParser*
  `public` **xParser**`( java.lang.String  full )`
  - **Usage**
    - Create a parser object and store string. String must have the format

      <Dns/Node[:NodeId]/[applNS::]Application[:ApplicationId]/Name >

      for standard parameters and

      <Dns/Name/[applNS::]Application[:ApplicationId]/Node[:NodeId] >

      for commands (only used for sorting purposes)
  - **Parameters**
    - `full` - full string to be parsed later.

Methods

- *buildCommand*
  `public void` **buildCommand**`( )`
  - **Usage**
    - build full string from items in command order

- *buildFull*
  `public void` **buildFull**`( )`
  - **Usage**
    - build full string from items

- *buildQuality*
  `public void` **buildQuality**`( int  s, int  t, int  v, int  m )`
  - **Usage**
    - store quality built from state, type, visibility, mode
  - **Parameters**
    - `s` - state: UpToDate, Unsolicited, Obsolete, Invalid, Undefined
    - `t` - type: Plain, State, Rate, Histogram, Module
    - `v` - visibility: Monitored, Changable
    - `m` - mode: not yet used

- *format*
  `public int` **format**`( )`
  - **Usage**
    - parse internal DIM format string set by setFormat(string). The items of the string are copied into internal string arrays and can be retreived by getter methods.

      DIM format strings have format

      <T:S;T:S;...>

      For scalar data set default size to 1 (eventually this should be 0?); for structures, the last item size defaults to 0 indicating that it has arbitrary size
  - **Returns** - 0 OK, -1 syntax error

- *format*
  `public int` **format**`( java.lang.String` **formstring**`, boolean` **store** `)`
  - **Usage**
    - parse given DIM format string. The items of the string are copied into internal string arrays and can be retreived by getter methods.

      DIM format strings have format

      <T:S;T:S;...>

      For scalar data set default size to 1 (eventually this should be 0?); for structures, the last item size defaults to 0 indicating that it has arbitrary size
  - **Parameters**
    - `formstring` - DIM format string
    - `store` - PARSE_STORE_FULL: store formstring, PARSE_ONLY: parse only
  - **Returns** - 0 OK, -1 syntax error

- *getApplication*
  `public String` **getApplication**`( )`
  - **Usage**
    - returns Application spec, which is name:port
  - **Returns** - Application spec, which is name:port

- *getApplicationFull*
  `public String` **getApplicationFull**`( )`
  - **Usage**
    - returns Application full, which is namespace::name:port
  - **Returns** - Application full, which is namespace::name:port

---

- *getApplicationID*
  `public String` **getApplicationID**`( )`

  - **Returns** - Application ID

- *getApplicationName*
`public String` **getApplicationName**`( )`

  - **Returns** - Application name

- *getCommand*
`public String` **getCommand**`( )`

  - **Returns** - internal full command string

- *getCommand*
`public String` **getCommand**`( boolean` **build** `)`

  - **Parameters**
    - `build` - MAKE_FULL: build string form fields and return, COPY_FULL: return current internal string
  - **Returns** - full string

- *getDns*
`public String` **getDns**`( )`

  - **Returns** - DNS

- *getFormat*
`public String` **getFormat**`( )`

  - **Returns** - formatstring

- *getFull*
`public String` **getFull**`( )`

  - **Returns** - internal full string

- *getFull*
`public String` **getFull**`( boolean` **build** `)`

  - **Parameters**
    - `build` - MAKE_FULL: build string form fields and return, COPY_FULL: return current internal string
  - **Returns** - full string

- *getItems*
`public String` **getItems**`( )`

  - **Returns** - item list (as separated by / in full name)

- *getMode*
`public int` **getMode**`( )`

  - **Usage**
    - get mode (from quality))
  - **Returns** - mode

- *getName*
  `public String` **getName**`( )`
  - **Returns** - parameter name

- *getNameSpace*
  `public String` **getNameSpace**`( )`
  - **Returns** - application namespace name

- *getNode*
  `public String` **getNode**`( )`
  - **Returns** - Node spec, which is name:port

- *getNodeID*
  `public String` **getNodeID**`( )`
  - **Returns** - Node ID

- *getNodeName*
  `public String` **getNodeName**`( )`
  - **Returns** - Node name

- *getNofTypes*
  `public int` **getNofTypes**`( )`
  - **Usage**
    - get number of types
  - **Returns** - Ntypes

- *getQuality*
  `public int` **getQuality**`( )`
  - **Usage**
    - get quality
  - **Returns** - quality

- *getState*
  `public int` **getState**`( )`
  - **Usage**
    - get state (from quality))
  - **Returns** - state

    NOTSPEC=0;

    UPTODATE=1;

    UNSOLICITED=2;

OBSOLETE=3;

INVALID=4;

UNDEFINED=5;

- *getType*
  `public int` **getType**`( )`
  - **Usage**
    - get structure type (from quality))
  - **Returns** - type

    ATOMIC=0;

    GENERIC=1;

    STATE=2;

    RATE=3;

    HISTOGRAM=4;

    MODULE=5;

    PORT=6;

    DEVICE=7;

    QUEUE=8;

- *getTypeList*
  `public String` **getTypeList**`( )`
  - **Usage**
    - get vector of types
  - **Returns** - types

- *getTypeName*
  `public String` **getTypeName**`( )`

- *getTypeSizes*
  `public int` **getTypeSizes**`( )`
  - **Usage**
    - get vector of sizes
  - **Returns** - sizes

- *getVisibility*
  `public int` **getVisibility**`( )`
  - **Usage**

    * get visibility (from quality))
  - **Returns** - visibility

    HIDDEN=0;

    MONITOR=1;

    CHANGABLE=2;

    IMPORTANT=4;

- *isArray*
  `public boolean` **isArray**`( )`

  - **Returns** - true if array

- *isAtomic*
  `public boolean` **isAtomic**`( )`

- *isChangable*
  `public boolean` **isChangable**`( )`

- *isChar*
  `public boolean` **isChar**`( )`

  - **Returns** - true if char

- *isCommandDescriptor*
  `public boolean` **isCommandDescriptor**`( )`

- *isDouble*
  `public boolean` **isDouble**`( )`

  - **Returns** - true if double

- *isError*
  `public boolean` **isError**`( )`

- *isFatal*
  `public boolean` **isFatal**`( )`

- *isFloat*
  `public boolean` **isFloat**`( )`

  - **Returns** - true if float

- *isGeneric*
  `public boolean` **isGeneric**`( )`

- *isHidden*
  `public boolean` **isHidden**`( )`

- *isHistogram*
  `public boolean` **isHistogram**`( )`

- *isImportant*
  `public boolean` **isImportant**`( )`

- *isInfo*
  `public boolean` **isInfo**`( )`

- *isInformation*
  `public boolean` **isInformation**`( )`

- *isInt*
  `public boolean` **isInt**`( )`
  - **Returns** - true if 32 bit int

- *isLogging*
  `public boolean` **isLogging**`( )`

- *isLong*
  `public boolean` **isLong**`( )`
  - **Returns** - true if 64 bit long

- *isMonitor*
  `public boolean` **isMonitor**`( )`

- *isNotSpecified*
  `public boolean` **isNotSpecified**`( )`

- *isRate*
  `public boolean` **isRate**`( )`

- *isState*
  `public boolean` **isState**`( )`

- *isStruct*
  `public boolean` **isStruct**`( )`
  - **Returns** - true if struct (all others false)

- *isSuccess*
  `public boolean` **isSuccess**`( )`

- *isVisible*
  `public boolean` **isVisible**`( )`

- *isWarning*
  `public boolean` **isWarning**`( )`

- *parse*
  `public int` **parse**`( )`
  - **Usage**
    - Parse internal full string which must have been set by constructor or setFull(string). The items of the string are copied into internal string items and can be retrieved by getter methods. return code:
  - **Returns** - 0 OK, -1 syntax error

- *parse*
  `public int` **parse(** `java.lang.String` **full,** `boolean` **store)**
  - **Usage**
    - Parse given full string. The items of the string are copied into internal strings. return code:
  - **Parameters**
    - `full` - string to parse. String must have the format

      <Dns/Node[:NodeId]/[applNS::]Application[:ApplicationId]/Name >
    - `store` - PARSE_STORE_FULL: store composed strings (parameter and command format), PARSE_ONLY: parse only
  - **Returns** - 0 OK, -1 syntax error

- *parse*
  `public int` **parse(** `java.lang.String` **full,** `boolean` **store,** `boolean` **command)**
  - **Usage**
    - Parse given full string. The items of the string are copied into internal strings.
  - **Parameters**
    - `full` - string String must have the format

      <Dns/Node[:NodeId]/[applNS::]Application[:ApplicationId]/Name >
    - `store` - PARSE_STORE_FULL: store composed strings (parameter and command format), PARSE_ONLY: parse only
    - `command` - IS_COMMAND: given string is command format

      <Dns/Name/[applNS::]Application[:ApplicationId]/Node[:NodeId] >

      IS_PARAMETER: standard
  - **Returns** - 0 OK, -1 syntax error

- *parseQuality*
  `public void` **parseQuality(** `int` **qual)**
  - **Usage**
    - store quality and parse into state, type, visibility, mode
  - **Parameters**
    - `qual` - DIM quality

- *printItems*
  `public void` **printItems(** **)**
  - **Usage**
    - Print all internal items

- *setApplicationID*
  `public void` **setApplicationID(** `java.lang.String` **applicationID)**
  - **Usage**

    * set application ID
  - **Parameters**
    * `applicationID` -

- *setApplicationName*
  `public void` **setApplicationName**`( java.lang.String` **application** `)`
  - **Usage**
    * set application name
  - **Parameters**
    * `application` -

- *setDns*
  `public void` **setDns**`( java.lang.String` **DNS** `)`
  - **Usage**
    * set DNS name
  - **Parameters**
    * `DNS` -

- *setFormat*
  `public void` **setFormat**`( java.lang.String` **formatstring** `)`
  - **Usage**
    * set DIM format string
  - **Parameters**
    * `formatstring` -

- *setFull*
  `public void` **setFull**`( java.lang.String` **full** `)`
  - **Usage**
    * set full string
  - **Parameters**
    * `full` -

- *setName*
  `public void` **setName**`( java.lang.String` **name** `)`
  - **Usage**
    * set parameter name
  - **Parameters**
    * `name` -

- *setNameSpace*
  `public void` **setNameSpace**`( java.lang.String` **name** `)`
  - **Usage**
    * set application namespace name
  - **Parameters**

* `name` -

- *setNodeID*
`public void` **setNodeID**`( java.lang.String` **nodeID** `)`
  - **Usage**
    * set node ID
  - **Parameters**
    * `nodeID` -

- *setNodeName*
`public void` **setNodeName**`( java.lang.String` **node** `)`
  - **Usage**
    * set node name
  - **Parameters**
    * `node` -

- *toString*
`public String` **toString**`( )`
  - **Returns** - full name

- *toString*
`public String` **toString**`( boolean` **command** `)`
  - **Parameters**
    * `command` - IS_COMMAND: return command format (name/appl/node), IS_PARAMETER: standard
  - **Returns** - full name in standard or command order

- *toString*
`public String` **toString**`( int` **indent**`, boolean` **command** `)`
  - **Parameters**
    * `indent` - Indentation level for browser: 0 DNS, 1 node (name for command), 2 application, 3 node (name for command).
    * `command` - IS_COMMAND: return command format (name/appl/node), IS_PARAMETER: standard
  - **Returns** - lettering depending on indent level

1.2.34 Class **xParTable**

Table model for parameter table.

Declaration

public class xParTable **extends** javax.swing.table.DefaultTableModel **implements** java.util.Comparator

CONSTRUCTORS

- *xParTable*
  `public` **xParTable**`( int` **edit**`, int` **meter** `)`

METHODS

- *addColumn*
  `public void` **addColumn**`( java.lang.Object` **columnName** `)`
  - **Usage**
    * addColumn methods are inherited from the DefaultTableModel class.

- *addColumn*
  `public void` **addColumn**`( java.lang.Object` **columnName**`, java.lang.Object []` **columnData** `)`

- *addColumn*
  `public void` **addColumn**`( java.lang.Object` **columnName**`, java.util.Vector` **columnData** `)`

- *addMouseListenerToHeaderInTable*
  `public void` **addMouseListenerToHeaderInTable**`( javax.swing.JTable` **table** `)`

- *compare*
  `public int` **compare**`( java.lang.Object` **v1**`, java.lang.Object` **v2** `)`
  - **Usage**
    * This method is the implementation of the Comparator interface. It is used for sorting the rows

- *getAscendingOrder*
  `public Integer` **getAscendingOrder**`( int` **c** `)`

- *getColumnClass*
  `public Class` **getColumnClass**`( int` **c** `)`

- *isCellEditable*
  `public boolean` **isCellEditable**`( int` **r**`, int` **c** `)`

- *sort*
  `public void` **sort**`( )`
  - **Usage**
    * This method sorts the rows using Java's Collections class. After sorting, it changes the info of the column - if the column was ascending, its new info is descending, and vice versa.

- *sortByColumn*
  `public void` **sortByColumn**`( int` **column** `)`

1.2.35 Class **xRate**

Graphic rate meter (bar). Can be used as item in xPanelGraphics.

Declaration

public class xRate
extends javax.swing.JPanel
implements xiPanelItem, java.awt.event.MouseListener, java.awt.event.ActionListener

Serializable Fields

Fields

- public static final int XSIZE
  - recommended size x
- public static final int YSIZE
  - recommended size y

Constructors

- *xRate*
  `public` **xRate**`( java.lang.String` **head**`, java.lang.String` **name**`, boolean` **header**`, int` **xlength**`, int` **ylength**`, double` **min**`, double` **max**`, java.lang.String` **color** `)`
  - **Usage**
    - Creates a Rate canvas.
  - **Parameters**
    - `head` - Name of parameter displayed.
    - `xlength` - Size of canvas in pixels.
    - `ylength` - Size of canvas in pixels.

Methods

- *actionPerformed*
  `public void` **actionPerformed**`( java.awt.event.ActionEvent` **a** `)`
- *getDimension*
  `public Dimension` **getDimension**`( )`

- *getHeader*
  `public String` **getHeader**`( )`
- *getID*
  `public int` **getID**`( )`
- *getMax*
  `public double` **getMax**`( )`
- *getMin*
  `public double` **getMin**`( )`
- *getName*
  `public String` **getName**`( )`
- *getPanel*
  `public JPanel` **getPanel**`( )`
- *getPosition*
  `public Point` **getPosition**`( )`
- *initRate*
  `public void` **initRate**`( java.lang.String` **head**`, java.lang.String` **name**`, int` **xlen**`, int` **ylen**`, double` **min**`, double` **max**`, java.lang.String` **color** `)`
  - **Usage**
    - Initializes a Rate canvas (called by constructor).
  - **Parameters**
    - `head` - Name of parameter displayed.
    - `xlen` - Size of canvas in pixels.
    - `ylen` - Size of canvas in pixels.
- *mouseClicked*
  `public void` **mouseClicked**`( java.awt.event.MouseEvent` **me** `)`
- *mouseEntered*
  `public void` **mouseEntered**`( java.awt.event.MouseEvent` **me** `)`
- *mouseExited*
  `public void` **mouseExited**`( java.awt.event.MouseEvent` **me** `)`
- *mousePressed*
  `public void` **mousePressed**`( java.awt.event.MouseEvent` **me** `)`
- *mouseReleased*
  `public void` **mouseReleased**`( java.awt.event.MouseEvent` **me** `)`
- *paintComponent*
  `public void` **paintComponent**`( java.awt.Graphics` **g** `)`
  - **Usage**
    - Called by repaint, calls update.
- *redraw*
  `public void` **redraw**`( )`
  - **Usage**

    * Redraw without changes (repaint).

- *redraw*
  `public void` **redraw**`( double` **value**`, boolean` **draw** `)`
  - **Usage**
    * Redraw with new value.
  - **Parameters**
    * `value` - New value.
    * `draw` - True for redraw.

- *setActionListener*
  `public void` **setActionListener**`( java.awt.event.ActionListener` **actionlistener** `)`

- *setColor*
  `public void` **setColor**`( java.lang.String` **colorname** `)`
  - **Usage**
    * Set color.
  - **Parameters**
    * `colorname` - (Red, Green, Blue, Yellow, Cyan, Magenta).

- *setColorBack*
  `public void` **setColorBack**`( java.awt.Color` **color** `)`

- *setID*
  `public void` **setID**`( int` **i** `)`

- *setSizeXY*
  `public void` **setSizeXY**`( )`

- *setSizeXY*
  `public void` **setSizeXY**`( java.awt.Dimension` **dd** `)`

- *update*
  `public void` **update**`( java.awt.Graphics` **g** `)`
  - **Usage**
    * Overwriting update method we avoid clearing the graphics. This would cause flickering update is not called by repaint!

1.2.36 Class **xRecord**

Base class for DIM record data.

Declaration

public class xRecord **extends** java.lang.Object

CONSTRUCTORS

- *xRecord*
  `public` **xRecord**`( java.lang.String` **name**`, int` **type** `)`
  - **Usage**
    - DIM record object:
  - **Parameters**
    - `name` - DABC format full parameter name
    - `type` - Type of record (like Meter, State, Info, Histo) as defined in parser
  - **See Also**
    - `xgui.xParser` (in 1.2.33, page 76)

METHODS

- *getColor*
  `public String` **getColor**`( )`
- *getName*
  `public String` **getName**`( )`
- *getPosition*
  `public Point` **getPosition**`( )`
- *getSize*
  `public Dimension` **getSize**`( )`
- *getType*
  `public int` **getType**`( )`
  - **Returns** - record type as defined in parser.
  - **See Also**
    - `xgui.xParser` (in 1.2.33, page 76)
- *isVisible*
  `public Boolean` **isVisible**`( )`
- *setColor*
  `public void` **setColor**`( java.lang.String` **color** `)`
  - **Usage**
    - DIM record object:
  - **Parameters**
    - `color` - Color depending on record.
- *setName*
  `public void` **setName**`( java.lang.String` **name** `)`
- *setPosition*
  `public void` **setPosition**`( java.awt.Point` **position** `)`
  - **Usage**

    * Set position. Normally the position is determined by the container panel like xPanelMeter. The position is relative to the container panel.
  - **Parameters**
    * `position` - Position of graphics.

- *setSize*
  `public void` **setSize**`( java.awt.Dimension` **dimension** `)`
  - **Usage**
    * Set size.
  - **Parameters**
    * `dimension` - Size of graphics.

- *setVisible*
  `public void` **setVisible**`( boolean` **Visible** `)`

- *setVisible*
  `public void` **setVisible**`( java.lang.String` **Visible** `)`
  - **Usage**
    * Set record visible

1.2.37 Class xRecordHisto

Dim record data for histogram.

Declaration

public class xRecordHisto **extends** xgui.xRecord

Constructors

- *xRecordHisto*
  `public` **xRecordHisto**`( java.lang.String` **name**`, int` **type**`, float` **lower**`, float` **upper**`, java.lang.String` **lettering**`, java.lang.String` **content**`, java.lang.String` **color** `)`
  - **Usage**
    * DIM record object: Histo
  - **Parameters**
    * `name` - DABC format full parameter name
    * `type` - Record type
    * `lower` - lower limit
    * `upper` - upper limit
    * `lettering` - Lettering of X axis.
    * `content` - Lettering of Y axis
    * `color` - color of pointer

METHODS

- *getLogScale*
  `public Boolean` **getLogScale**`( )`
- *getLower*
  `public float` **getLower**`( )`
- *getMode*
  `public int` **getMode**`( )`
- *getUpper*
  `public float` **getUpper**`( )`
- *getValue*
  `public int` **getValue**`( )`
- *printRecord*
  `public void` **printRecord**`( )`
- *restoreRecord*
  `public void` **restoreRecord**`( org.w3c.dom.Element` **el** `)`
- *setContent*
  `public void` **setContent**`( java.lang.String` **content** `)`
- *setLettering*
  `public void` **setLettering**`( java.lang.String` **lettering** `)`
- *setLogScale*
  `public void` **setLogScale**`( java.lang.Boolean` **log** `)`
- *setLower*
  `public void` **setLower**`( float` **low** `)`
- *setMode*
  `public void` **setMode**`( int` **mode** `)`
- *setUpper*
  `public void` **setUpper**`( float` **up** `)`
- *setValue*
  `public void` **setValue**`( int` **ich**`, int []` **iar** `)`
- *XmlLine*
  `public String` **XmlLine**`( )`

1.2.38 CLASS xRecordInfo

Dim record data for info.

DECLARATION

public class xRecordInfo **extends** xgui.xRecord

CONSTRUCTORS

- *xRecordInfo*
  `public` **xRecordInfo**`( java.lang.String` **name**`, int` **type** `)`
  - **Usage**
    - DIM record object: Info
  - **Parameters**
    - `name` - DABC format full parameter name
    - `type` - Record type

METHODS

- *getSeverity*
  `public int` **getSeverity**`( )`
- *getValue*
  `public String` **getValue**`( )`
- *setValue*
  `public void` **setValue**`( int` **severity**`, java.lang.String` **color**`, java.lang.String` **value** `)`
  - **Usage**
    - DIM record object: Info
  - **Parameters**
    - `severity` -
    - `color` - color of text.
    - `value` - Text

1.2.39 CLASS xRecordMeter

Dim record data for meter.

DECLARATION

public class xRecordMeter
extends xgui.xRecord

CONSTRUCTORS

- *xRecordMeter*
  `public` **xRecordMeter**`( java.lang.String` **name**`, int` **type**`, int` **mode**`, double` **lower**`, double` **upper**`, double` **alarmLower**`, double` **alarmUpper**`, java.lang.String` **color**`, java.lang.String` **alarmColor**`, java.lang.String` **units** `)`

- **Usage**
  - DIM record object: Meter
- **Parameters**
  - `name` - DABC format full parameter name
  - `type` - Record type
  - `mode` - Display mode
  - `lower` - lower limit
  - `upper` - upper limit
  - `alarmLower` - alarm lower limit
  - `alarmUpper` - alarm upper limit
  - `color` - color of pointer
  - `alarmColor` - color of frame in alarm
  - `units` - units

METHODS

- *getAutoScale*
  `public Boolean` **getAutoScale**`( )`
- *getLogScale*
  `public Boolean` **getLogScale**`( )`
- *getLower*
  `public double` **getLower**`( )`
- *getMode*
  `public int` **getMode**`( )`
- *getUnits*
  `public String` **getUnits**`( )`
- *getUpper*
  `public double` **getUpper**`( )`
- *getValue*
  `public double` **getValue**`( )`
- *printRecord*
  `public void` **printRecord**`( )`
- *restoreRecord*
  `public void` **restoreRecord**`( org.w3c.dom.Element` **el** `)`
- *setAutoScale*
  `public void` **setAutoScale**`( java.lang.Boolean` **auto** `)`
- *setLogScale*
  `public void` **setLogScale**`( java.lang.Boolean` **log** `)`
- *setLower*
  `public void` **setLower**`( double` **low** `)`
- *setMode*
  `public void` **setMode**`( int` **mode** `)`

- *setUpper*
  `public void` **setUpper**`( double` **up** `)`
- *setValue*
  `public void` **setValue**`( double` **value** `)`
- *XmlLine*
  `public String` **XmlLine**`( )`

1.2.40 Class xRecordState

Dim record data for state.

Declaration

public class xRecordState
extends xgui.xRecord

Constructors

- *xRecordState*
  `public` **xRecordState**`( java.lang.String` **name**`, int` **type** `)`
  - **Usage**
    - DIM record object: Meter
  - **Parameters**
    - `name` - DABC format full parameter name
    - `type` - Record type

Methods

- *getSeverity*
  `public int` **getSeverity**`( )`
- *getValue*
  `public String` **getValue**`( )`
- *setValue*
  `public void` **setValue**`( int` **severity**`, java.lang.String` **color**`,`
  `java.lang.String` **value** `)`

1.2.41 Class xRemoteShell

Remote shell execution.

DECLARATION

public class xRemoteShell
extends java.lang.Object

CONSTRUCTORS

- *xRemoteShell*
  `public` **xRemoteShell**`( java.lang.String` **shell** `)`

METHODS

- *dir*
  `public void` **dir**`( java.lang.String` **node**`, java.lang.String` **user**`, java.lang.String` **cmd** `)`
- *rsh*
  `public boolean` **rsh**`( java.lang.String` **node**`, java.lang.String` **user**`, java.lang.String` **cmd**`, long` **timeout** `)`
- *rshout*
  `public String` **rshout**`( java.lang.String` **node**`, java.lang.String` **user**`, java.lang.String` **cmd** `)`

1.2.42 CLASS xSaveRestore

Base class for DIM SaveRestore data.

DECLARATION

public class xSaveRestore
extends java.lang.Object

CONSTRUCTORS

- *xSaveRestore*
  `public` **xSaveRestore**`( )`
  - **Usage**
    - DIM SaveRestore object:

METHODS

- *restoreLayouts*
  `protected static final void` **restoreLayouts**`( java.lang.String` **file** `)`
  - **Usage**
    - Restore layouts from xml file
  - **Parameters**
    - `file` - File name (ending with .xml).
  - **See Also**
    - `xgui.xSet` (in 1.2.43, page 100)

- *restoreRecords*
  `public static final void` **restoreRecords**`( java.lang.String` **file** `)`
  - **Usage**
    - Restore all records attached to parameters (meters and histograms). XML tree is stored in xSet and can be retrieved by getRecordXml called in xPanelParameter.
  - **Parameters**
    - `file` - Xml file name.
  - **See Also**
    - `xgui.xSet` (in 1.2.43, page 100)
    - `xgui.xPanelParameter` (in 1.2.28, page 67)

- *saveLayouts*
  `protected static final void` **saveLayouts**`( java.lang.String` **file** `)`
  - **Usage**
    - Save layouts to xml file
  - **Parameters**
    - `file` - File name (ending with .xml).
  - **See Also**
    - `xgui.xSet` (in 1.2.43, page 100)

- *saveRecords*
  `public static final void` **saveRecords**`( java.util.Vector` **vpar**`, java.lang.String` **file** `)`
  - **Usage**
    - Save all records attached to parameters (meters and histograms).
  - **Parameters**
    - `vpar` - Vector of xiDimParameters.
    - `file` - Xml file name.

1.2.43 CLASS xSet

Singleton and registry.

Declaration

public class xSet
extends java.lang.Object

Fields

- public static boolean OPEN
  –
- public static boolean CLOSE
  –

Constructors

- *xSet*
  `public` **xSet**`( )`
  - **Usage**
    * Singleton

Methods

- *addDimension*
  `public static final Dimension` **addDimension**`( java.awt.Dimension` **d1**`, java.awt.Dimension` **d2** `)`
  - **Usage**
    * Dimension adder for convenience.
  - **Parameters**
    * `d1` - First dimension
    * `d2` - Second dimension
  - **Returns** - New dimension with added w and h.

- *addObject*
  `public static final Object` **addObject**`( java.lang.Object` **object** `)`
  - **Usage**
    * Adds object to repository (only one per class!).
  - **Parameters**
    * `object` - Object to add.
  - **Returns** - Null if an object of this class already exists.

- *black*
  `public static final Color` **black**`( )`

- *blue*
  `public static final Color` **blue**`( )`

- *blueD*
  `public static final Color` **blueD**`( )`

- *blueL*
  `public static final Color` **blueL**`( )`

- *createLayout*
  `public static final xLayout` **createLayout**`( java.lang.String` **name** `)`
  - **Usage**
    - Layouts keep position and size of windows. Most layouts are created in xDesktop, but user can add his own. All layouts are stored with the save layout button. On startup of the GUI layouts are retrieved. User can get his layouts by getLayout method.
  - **Parameters**
    - `name` - Name of layout.
  - **Returns** - New layout or null, if one already exists with same name.

- *createLayout*
  `public static final xLayout` **createLayout**`( java.lang.String` **name**`, java.awt.Point` **pos**`, java.awt.Dimension` **size**`, int` **columns**`, boolean` **visible** `)`
  - **Usage**
    - See also createLayout.
  - **Parameters**
    - `name` - Name of layout.
    - `pos` - Position
    - `size` - Size of window
    - `columns` - Columns in the panels of states, meters, and histograms.
    - `visible` - True if component should be show up on startup.
  - **Returns** - New layout or null, if one already exists with same name.

- *cyan*
  `public static final Color` **cyan**`( )`

- *cyanD*
  `public static final Color` **cyanD**`( )`

- *cyanL*
  `public static final Color` **cyanL**`( )`

- *getAccess*
  `public static final String` **getAccess**`( )`
  - **Returns** - Encrypted password.

- *getColorBack*
  `public static final Color` **getColorBack**`( )`

- *getColorText*
  `public static final Color` **getColorText**`( )`

- *getDesktop*
  `protected static final JDesktopPane` **getDesktop**`( )`
  - **Usage**
    - Sometimes one needs the top pane.
  - **Returns** - Desktop pane.
  - **See Also**
    - `xgui.xHisto` (in 1.2.12, page 34)
    - `xgui.xMeter` (in 1.2.18, page 48)
    - `xgui.xPanelPrompt` (in 1.2.29, page 69)

- *getDimDns*
  `public static final String` **getDimDns**`( )`
  - **Returns** - DIM name server node

- *getGuiNode*
  `public static final String` **getGuiNode**`( )`
  - **Returns** - Host name.

- *getIcon*
  `public static final ImageIcon` **getIcon**`( java.lang.String` **file** `)`
  - **Usage**
    - To read images from jar file, we need this ugly stuff.
  - **Parameters**
    - `file` - Icon file name.
  - **Returns** - The icon or null (not found).

- *getLayout*
  `public static final xLayout` **getLayout**`( java.lang.String` **name** `)`
  - **Usage**
    - See also createLayout.
  - **Parameters**
    - `name` - Name of layout.
  - **Returns** - Layout or null (not found).

- *getLayouts*
  `public static final Vector` **getLayouts**`( )`
  - **Usage**
    - See also createLayout.
  - **Returns** - Layouts.

- *getMessage*
  `public static final String` **getMessage**`( )`

  - **Returns** - Value previously set by setMessage(...).

- *getObject*
  `public static final Object` **getObject**`( java.lang.String` **classname** `)`
  - **Parameters**
    - `classname` - Only one object per class!
  - **Returns** - Object reference or null.

- *getParTableWidth*
  `protected static final int` **getParTableWidth**`( )`
  - **Returns** - Widths of the parameter table columns.

- *getRecordXml*
  `public static final NodeList` **getRecordXml**`( )`

- *getUserName*
  `public static final String` **getUserName**`( )`
  - **Returns** - User name.

- *gray*
  `public static final Color` **gray**`( )`

- *grayD*
  `public static final Color` **grayD**`( )`

- *grayL*
  `public static final Color` **grayL**`( )`

- *green*
  `public static final Color` **green**`( )`

- *greenD*
  `public static final Color` **greenD**`( )`

- *greenL*
  `public static final Color` **greenL**`( )`

- *isSuccess*
  `public static final boolean` **isSuccess**`( )`
  - **Returns** - Value previously set by setSuccess(...).

- *magenta*
  `public static final Color` **magenta**`( )`

- *magentaD*
  `public static final Color` **magentaD**`( )`

- *magentaL*
  `public static final Color` **magentaL**`( )`

- *readXml*
  `public static final Element` **readXml**`( java.lang.String` **file** `)`

  - **Usage**
    * Read XML file
  - **Parameters**
    * `file` - File name (ending with .xml).
  - **Returns** - Top element.

- *red*
  `public static final Color` **red**`( )`

- *redD*
  `public static final Color` **redD**`( )`

- *redL*
  `public static final Color` **redL**`( )`

- *setAccess*
  `protected static final void` **setAccess**`( char [] access )`

  - **Usage**
    * Store encrypted password (crypt).
  - **Parameters**
    * `access` - Password as retrieved from JPasswordField.

- *setColorBack*
  `public static final void` **setColorBack**`( java.awt.Color` **back** `)`

  - **Usage**
    * Set background color. Userpanel may set this color in constructor. Than all panels inherit this color.
  - **Parameters**
    * `back` - Color for background

- *setColorText*
  `public static final void` **setColorText**`( java.awt.Color` **text** `)`

- *setDefaultCursor*
  `protected static final void` **setDefaultCursor**`( )`

  - **Usage**
    * Does not work as expected!

- *setDesktop*
  `protected static final void` **setDesktop**`( javax.swing.JFrame` **gui**`, javax.swing.JDesktopPane` **dp** `)`

- *setLayout*
  `public static final boolean` **setLayout**`( java.lang.String` **name**`, java.awt.Point` **pos**`, java.awt.Dimension` **size**`, int` **columns**`, boolean` **visible** `)`

  - **Usage**
    * See also createLayout.
  - **Parameters**

    * `name` - Name of layout.
    * `pos` - Position
    * `size` - Size of window
    * `columns` - Columns in the panels of states, meters, and histograms.
    * `visible` - True if component should be show up on startup.
  - **Returns** - True if layout exists, otherwise null.
  - **See Also**
    * `xgui.xLayout` (in 1.2.16, page 46)

- *setMessage*
  `public static final void` **setMessage**`( java.lang.String` **message** `)`
  - **Parameters**
    * `message` - Value can be retrieved by getMessage().

- *setParTableWidth*
  `protected static final void` **setParTableWidth**`( int [] w )`

- *setRecordXml*
  `public static final void` **setRecordXml**`( org.w3c.dom.NodeList` **list** `)`

- *setSuccess*
  `public static final void` **setSuccess**`( boolean` **success** `)`
  - **Parameters**
    * `success` - Value can be retrieved by isSuccess().

- *setTableLayout*
  `protected static final void` **setTableLayout**`( java.lang.String` **name**`,`
  `org.w3c.dom.Element` **layout** `)`
  - **Usage**
    * Set table values from XML element. Called by xSaveRestore.restoreLayouts.
  - **Parameters**
    * `name` - Name of element of TableLayout ("Parameter").
    * `layout` - Element of TableLayout.
  - **See Also**
    * `xgui.xSaveRestore` (in 1.2.42, page 99)

- *setWaitCursor*
  `protected static final void` **setWaitCursor**`( )`
  - **Usage**
    * Does not work as expected!

- *setWindowLayout*
  `protected static final void` **setWindowLayout**`( org.w3c.dom.Element` **layout** `)`
  - **Usage**
    * Set layout values from XML element. Called by xSaveRestore.restoreLayouts.
  - **Parameters**

    - `layout` - Element of WindowLayout elements.
  - **See Also**
    - `xgui.xSaveRestore` (in 1.2.42, page 99)

- *white*
  `public static final Color` **white**`( )`

- *writeXml*
  `public static final void` **writeXml**`( java.lang.String` **file**`, java.lang.String` **xml** `)`
  - **Usage**
    - Write xml file
  - **Parameters**
    - `file` - File name (ending with .xml).
    - `xml` - Xml string to write.

- *XmlHeader*
  `public static final String` **XmlHeader**`( )`
  - **Usage**
    - XML opening string.
  - **Returns** - Standard XML opening string.

- *XmlTag*
  `public static final String` **XmlTag**`( java.lang.String` **name**`, boolean` **open** `)`
  - **Usage**
    - XML tag string.
  - **Parameters**
    - `name` - Tag name.
    - `open` - True: open tag, false: close tag.
  - **Returns** - XML string.

- *yellow*
  `public static final Color` **yellow**`( )`

- *yellowD*
  `public static final Color` **yellowD**`( )`

- *yellowL*
  `public static final Color` **yellowL**`( )`

1.2.44 Class **xSetup**

Used in DABC form panel to read/edit/write Xdaq setup files.

Declaration

public class xSetup
extends java.lang.Object

Constructors

- *xSetup*
  `public` **xSetup**`( )`

Methods

- *getContextNumber*
  `protected int` **getContextNumber**`( )`
  - **Returns** - Number of contexts found.

- *getContexts*
  `protected NodeList` **getContexts**`( )`
  - **Returns** - List of contexts.

- *getNames*
  `protected Vector` **getNames**`( )`
  - **Returns** - Flat list of names (reference).

- *getTypes*
  `protected Vector` **getTypes**`( )`
  - **Returns** - Flat list of types (reference).

- *getValues*
  `protected Vector` **getValues**`( )`
  - **Returns** - Flat list of values (reference).

- *parseSetup*
  `protected boolean` **parseSetup**`( java.lang.String` **file** `)`
  - **Usage**
    - Read and parse Xdaq XML setup file. Generates flat lists of names/types/values. Loops over "xc:Context", then "xc:Module" and "xc:Application"+"properties"+"*". Assumption is that there is only one application per context.
  - **Parameters**
    - `file` - XML file name.
  - **See Also**
    - `xgui.xPanelDabc` (in 1.2.20, page 55)

- *printSetup*
  `public void` **printSetup**`( )`
  - **Usage**
    - Print flat list of names and values and types.

- *updateSetup*
  `protected boolean` **updateSetup**`( )`
  - **Usage**
    - Rebuild the XML values from values in the flat list. Caller gets references to these lists and may change values.
  - **See Also**
    - `xgui.xPanelDabc` (in 1.2.20, page 55)
    - `xgui.xPanelSetup` (in 1.2.31, page 73)

- *writeSetup*
  `protected boolean` **writeSetup**`( java.lang.String` **filename** `)`
  - **Usage**
    - Write internal XML string to XML file.
  - **Parameters**
    - `filename` - File name.
  - **Returns** - True: OK, false: error.

1.2.45 Class xState

Graphic item state.

Declaration

public class xState
extends javax.swing.JPanel
implements xiPanelItem, java.awt.event.MouseListener, java.awt.event.ActionListener

Serializable Fields

Fields

- public static final int XSIZE
  - normal size x
- public static final int YSIZE
  - normal size y

Constructors

- *xState*
  `public` **xState**`( java.lang.String` **head**`, int` **xlength**`, int` **ylength** `)`
  - **Usage**
    - Creates a State canvas.
  - **Parameters**
    - `head` - Name of parameter displayed.
    - `xlength` - Size of canvas in pixels.
    - `ylength` - Size of canvas in pixels.

Methods

- *actionPerformed*
  `public void` **actionPerformed**`( java.awt.event.ActionEvent` **a** `)`
- *getDimension*
  `public Dimension` **getDimension**`( )`
- *getID*
  `public int` **getID**`( )`
- *getName*
  `public String` **getName**`( )`
- *getPanel*
  `public JPanel` **getPanel**`( )`
- *getPosition*
  `public Point` **getPosition**`( )`
- *initState*
  `public void` **initState**`( java.lang.String` **head**`, int` **xlen**`, int` **ylen** `)`
  - **Usage**
    - Initializes a State canvas (called by constructor).
  - **Parameters**
    - `head` - Name of parameter displayed.
    - `xlen` - Size of canvas in pixels.
    - `ylen` - Size of canvas in pixels.
- *mouseClicked*
  `public void` **mouseClicked**`( java.awt.event.MouseEvent` **me** `)`
- *mouseEntered*
  `public void` **mouseEntered**`( java.awt.event.MouseEvent` **me** `)`
- *mouseExited*
  `public void` **mouseExited**`( java.awt.event.MouseEvent` **me** `)`
- *mousePressed*
  `public void` **mousePressed**`( java.awt.event.MouseEvent` **me** `)`

- *mouseReleased*
  `public void` **mouseReleased**`( java.awt.event.MouseEvent` **me** `)`

- *paintComponent*
  `public void` **paintComponent**`( java.awt.Graphics` **g** `)`
  - **Usage**
    - Called by repaint, calls update.

- *redraw*
  `public void` **redraw**`( )`
  - **Usage**
    - Redraw without changes (repaint).

- *redraw*
  `public void` **redraw**`( int` **severity**`, java.awt.Color` **color**`, java.lang.String` **value**`, boolean` **draw** `)`
  - **Usage**
    - Redraw with new value, calls redraw.
  - **Parameters**
    - `severity` - (0: Success, 1: Warning, 2: Error, 3: Fatal)
    - `color` - Color.
    - `value` - Short string describing state.
    - `draw` - True: redraw, false: update values only.

- *redraw*
  `public void` **redraw**`( int` **severity**`, java.lang.String` **colorname**`, java.lang.String` **value**`, boolean` **draw** `)`
  - **Usage**
    - Redraw with new value, calls redraw.
  - **Parameters**
    - `severity` - (0: Success, 1: Warning, 2: Error, 3: Fatal)
    - `colorname` - (Red, Green, Blue, Yellow, Cyan, Magenta).
    - `value` - Short string describing state.
    - `draw` - True: redraw, false: update values only.

- *setActionListener*
  `public void` **setActionListener**`( java.awt.event.ActionListener` **actionlistener** `)`

- *setColor*
  `public void` **setColor**`( java.awt.Color` **color** `)`
  - **Usage**
    - Set color.
  - **Parameters**
    - `color` - Color.

- *setColor*
  `public void` **setColor**`( java.lang.String` **colorname** `)`

  - **Usage**
    * Set color.
  - **Parameters**
    * `colorname` - (Red, Green, Blue, Yellow, Cyan, Magenta).

- *setColorBack*
  `public void` **setColorBack**`( java.awt.Color` **color** `)`
- *setID*
  `public void` **setID**`( int` **i** `)`
- *setSizeXY*
  `public void` **setSizeXY**`( )`
- *setSizeXY*
  `public void` **setSizeXY**`( java.awt.Dimension` **dd** `)`
- *showText*
  `public void` **showText**`( boolean` **show** `)`
- *update*
  `public void` **update**`( java.awt.Graphics` **g** `)`
  - **Usage**
    * Overwriting update method we avoid clearing the graphics. This would cause flickering update is not called by repaint!

1.2.46 Class xTimer

Timer class to launch actions.

Declaration

public class xTimer
extends javax.swing.Timer

Constructors

- *xTimer*
  `public` **xTimer**`( java.awt.event.ActionListener` **actlis**`, boolean` **repeat** `)`
  - **Usage**
    * Constructor of xTimer.
  - **Parameters**
    * `actlis` - Passed to Timer.
    * `repeat` - True: repeat firing the timer, false: only once

METHODS

- *action*
  `public void` **action**`( java.awt.event.ActionEvent` **aev** `)`
  - **Usage**
    - Call fireActionPerformed function of Timer (could be called directly instead).
  - **Parameters**
    - `aev` - Event passed to the fireActionPerformed function of Timer.

1.2.47 CLASS xXmlParser

Parser for XML formatted command descriptions.

DECLARATION

public class xXmlParser
extends java.lang.Object

CONSTRUCTORS

- *xXmlParser*
  `public` **xXmlParser**`( )`
  - **Usage**
    - The XML parser can be used to build a command description string. Or it can parse a given XML string and return the fields.

METHODS

- *addArgument*
  `public void` **addArgument**`( java.lang.String` **argument**`, java.lang.String` **type**`, java.lang.String` **value**`, java.lang.String` **required** `)`
  - **Usage**
    - Adds argument description to internal XML string buffer.
      argument name="" type="" value="" required=""
  - **Parameters**
    - `argument` - Name of argument
    - `type` - Type of argument (I,F,D,C)
    - `value` - Value of argument. In command definition would be the default.
    - `required` - Specifies if argument is required. Stored as string "true" or "false".

- *getArgumentName*
  `public String` **getArgumentName**`( int` **index** `)`
  - **Parameters**

    * `index` - Argument index.
  - **Returns** - Argument name field

- *getArgumentRequired*
  `public String` **getArgumentRequired**`( int` **index** `)`
  - **Parameters**
    * `index` - Argument index.
  - **Returns** - Argument required field (FALSE or TRUE)

- *getArgumentType*
  `public String` **getArgumentType**`( int` **index** `)`
  - **Parameters**
    * `index` - Argument index.
  - **Returns** - Argument type field(I,F,D,C)

- *getArgumentValue*
  `public String` **getArgumentValue**`( int` **index** `)`
  - **Parameters**
    * `index` - Argument index.
  - **Returns** - Argument value field

- *getCommandContent*
  `public String` **getCommandContent**`( )`
  - **Returns** - Command content field

- *getCommandName*
  `public String` **getCommandName**`( )`
  - **Returns** - Command name field

- *getCommandScope*
  `public String` **getCommandScope**`( )`
  - **Returns** - Command scope field

- *getName*
  `public String` **getName**`( )`
  - **Returns** - Component name

- *getNargs*
  `public int` **getNargs**`( )`
  - **Returns** - Number of arguments

- *getXmlString*
  `public String` **getXmlString**`( )`
  - **Returns** - Finalized internal XML string buffer, or XML string read by parseXmlString.

- *isChanged*
  `public boolean` **isChanged**`( )`
  - **Returns** - True: XML string contains values of arguments.

- *newCommand*
  `public void` **newCommand**`( java.lang.String` **command**`, boolean` **header** `)`
  - **Usage**
    - Starts to build internal XML string buffer.
      command name="" scope="" content="default"
  - **Parameters**
    - `command` - Name of the command
    - `header` - True: write XML header line (version, encoding).

- *newCommand*
  `public void` **newCommand**`( java.lang.String` **command**`, boolean` **header**`,`
  `java.lang.String` **scope** `)`
  - **Usage**
    - Starts to build internal XML string buffer.
      command name="" scope="" content="default"
  - **Parameters**
    - `command` - Name of the command
    - `header` - True: write XML header line (version, encoding).
    - `scope` - Scope of command, could be like public, hidden, MBS ...

- *newCommand*
  `public void` **newCommand**`( java.lang.String` **command**`, boolean` **header**`,`
  `java.lang.String` **scope**`, boolean` **changed** `)`
  - **Usage**
    - Starts to build internal XML string buffer.
      command name="" scope="" content="default"
  - **Parameters**
    - `command` - Name of the command
    - `header` - True: write XML header line (version, encoding).
    - `scope` - Scope of command, could be like public, hidden, ...
    - `changed` - False: content="default", true: content="values"

- *parseXmlString*
  `public void` **parseXmlString**`( java.lang.String` **name**`, java.lang.String`
  xmlstring `)`
  - **Usage**
    - Read and parse XML string. String is stored. Must be called before the getter methods.
  - **Parameters**
    - `name` - Just a name returned by getName.
    - `xmlstring` - Encoded XML string.

- *saveXmlString*
  `public void` **saveXmlString**`( java.lang.String` **file**`, java.lang.String` **xml** `)`
  - **Usage**
    - Save string to file.
  - **Parameters**
    - `file` - File name.
    - `xml` - XML string.