

# DockingFrames 1.0.8 - Transition

Benjamin Sigg

September 26, 2010

## Contents

<b>1</b>	<b>Version 1.0.3</b>	<b>6</b>
1.1	Incompatibilities . . . . .	6
1.1.1	DefaultKeyboardController . . . . .	6
1.1.2	DefaultDockable/DefaultCDockable . . . . .	6
1.1.3	CDockableListener . . . . .	6
1.1.4	FlapDockStation . . . . .	7
1.1.5	XML . . . . .	7
1.1.6	DockTheme . . . . .	7
1.1.7	DockFactory . . . . .	7
1.2	Features . . . . .	7
1.2.1	SplitDockStation . . . . .	7
1.2.2	SplitLayoutManager . . . . .	8
1.2.3	CDockable resize lock . . . . .	8
1.2.4	FlapLayoutManager . . . . .	8
1.2.5	ColorManager/ColorScheme . . . . .	8
1.2.6	ColorMap . . . . .	8
1.2.7	LookAndFeel . . . . .	8
1.2.8	CDockable resize request . . . . .	8
<b>2</b>	<b>Version 1.0.4</b>	<b>9</b>
2.1	Incompatibilities . . . . .	9
2.1.1	Binary file format . . . . .	9
2.1.2	DockableListener . . . . .	9
2.1.3	Title visibility on CDockables . . . . .	9
2.1.4	BasicDropDownButtonHandler . . . . .	9
2.1.5	CDockable.getClose . . . . .	9
2.1.6	CLocation . . . . .	10
2.1.7	working area . . . . .	10
2.2	Features . . . . .	10
2.2.1	Border around BubbleDisplayer . . . . .	10
2.2.2	Backup factories (core) . . . . .	10
2.2.3	Backup factories (common) . . . . .	10
2.2.4	Unregister factories from DockFrontend . . . . .	10
2.2.5	Action support keyboard . . . . .	11
2.2.6	FocusTraversalPolicies . . . . .	11
2.2.7	override predefined actions . . . . .	11

2.2.8	CBlank	11
2.2.9	CStation	11
2.3	Bugfixes	11
2.3.1	BubbleDisplayer.getDockableInsets	11
2.3.2	IndexOutOfBoundsException from ButtonPanel	11
2.3.3	Mode change of CDockable	12
2.3.4	Opening maximized CDockable	12
2.3.5	Unbind of DockAction called to often	12
<b>3</b>	<b>Version 1.0.5</b>	<b>12</b>
3.1	Incompatibilities	12
3.1.1	DockStationListener	12
3.1.2	DockableFocusListener	13
3.1.3	DockTheme.getDockableSelection	13
3.1.4	tap-strip no longer painted by TapPainter	13
3.1.5	KeyboardController does fire less events	13
3.1.6	ComponentHierarchyObserver	13
3.2	API and Layout	14
3.2.1	KeyStroke for closing Dockable	14
3.2.2	New listeners	14
3.2.3	ComponentHierarchObserver	14
3.2.4	Root window for DockController	14
3.2.5	FocusTraversalPolicies	14
3.2.6	Dialog to select focused Dockable	14
3.2.7	Extracting colors from LookAndFeel	15
3.2.8	EclipseTheme	15
3.2.9	SplitDockStation	15
3.3	Bugfixes	15
3.3.1	Missing colors for BasicTheme	15
3.3.2	Cutting bounds of children of SplitDockStation	15
3.3.3	NullPointerException when changing focus	15
3.3.4	Undecorated dialogs not undecorated	15
3.3.5	RexTabbedComponent not adding/removing children	16
3.3.6	Focusing a hidden CDockable	16
3.3.7	Missing events when changing state of CDockable	16
<b>4</b>	<b>Version 1.0.6</b>	<b>16</b>
4.1	Incompatibilities	16
4.1.1	Dockable with Tooltip	16
4.1.2	ColorManager generalized	16
4.1.3	Resize Request in Common	17
4.1.4	DockElementRepresentative	17
4.1.5	SimpleModifierMask deleted	17
4.1.6	Map of DockThemes	17
4.1.7	Persistent storage of DockTheme	18
4.2	API and Layout	18
4.2.1	Dropping onto SplitDockStation	18
4.2.2	UIProperties	18
4.2.3	Opened LockedResizeLayoutManager	18
4.2.4	ConflictResolver for locked resize	18

4.2.5	FullLockConflictResolver . . . . .	18
4.2.6	DockElementRepresentative . . . . .	19
4.2.7	Common: close-action and setVisible . . . . .	19
4.2.8	Preference system . . . . .	19
4.2.9	ColorScheme as property . . . . .	19
4.2.10	Default locations in Common . . . . .	19
4.2.11	Borders on OverpaintablePanel . . . . .	19
4.2.12	SplitDockStation can disabled resizing . . . . .	19
4.2.13	Handle AWT components . . . . .	20
4.3	Bugfixes . . . . .	20
4.3.1	DefaultConflictResolver did not respect locked sizes . . . . .	20
4.3.2	Opening maximized CDockable . . . . .	20
4.3.3	Dropping Dockable on SplitDockStation . . . . .	20
4.3.4	CSplitLocation broken . . . . .	20
4.3.5	CStateManager.getLocation broken . . . . .	20
4.3.6	Stack-component of EclipseTheme broken . . . . .	20
4.3.7	Change ColorScheme could throw NPE . . . . .	20
4.3.8	Items in popup-menu did do nothing . . . . .	21
<b>5</b>	<b>Version 1.0.7</b>	<b>21</b>
5.1	Incompatibilities . . . . .	21
5.1.1	DockableDisplayerHints . . . . .	21
5.1.2	ScreenDockDialog extends new class . . . . .	21
5.1.3	DockFactory uses a new layer . . . . .	21
5.1.4	CGridArea implements CDockable . . . . .	21
5.1.5	CWorkingArea extends CGridArea . . . . .	22
5.1.6	CControlFactory: no longer creates CWorkingAreas . . . . .	22
5.1.7	CommonDockable: getClose replaced with getSources . . . . .	22
5.1.8	PropertyKey: requires factory for default value . . . . .	22
5.2	API and Layout . . . . .	22
5.2.1	Button-title supports colors . . . . .	22
5.2.2	FlapDockProperty: support state and size . . . . .	23
5.2.3	AdjacentDockFactory . . . . .	23
5.2.4	DockSituation: support for missing elements . . . . .	23
5.2.5	Support for gaps in layout . . . . .	23
5.2.6	Storing information of invisible dockables . . . . .	23
5.2.7	Access to information of missing dockable . . . . .	23
5.2.8	PreferenceTable: order of operations reversed . . . . .	23
5.2.9	Automatic stack creation in CGrid . . . . .	24
5.2.10	Central register for CDockables . . . . .	24
5.2.11	FontManager . . . . .	24
5.2.12	FontMap . . . . .	24
5.2.13	More than one maximize-area . . . . .	24
5.2.14	Veto before changing layout . . . . .	24
5.2.15	CGrid/SplitDockGrid: preselect element . . . . .	24
5.2.16	WindowProviders . . . . .	25
5.2.17	AppletWindowProvider . . . . .	25
5.2.18	LocaleListener . . . . .	25
5.2.19	DockController: freeze layout . . . . .	25
5.3	Bugfixes . . . . .	25

5.3.1	SplitDockStation not respecting acceptances . . . . .	25
5.3.2	Common and CGrid: not supporting big stacks . . . . .	25
5.3.3	DockFrontend did not read setting correctly . . . . .	25
5.3.4	Infinite recursion in focus raversal . . . . .	25
5.3.5	CWorkingArea not settings itself as working-area . . . . .	26
5.3.6	SplitDockGrid throwing exception . . . . .	26
5.3.7	FlapWindow not resizing . . . . .	26
5.3.8	Dropping CWorkingArea . . . . .	26
5.3.9	Drop CDockable with no location but working-area . . . . .	26
5.3.10	AbstractCDockable ignores settings . . . . .	26
5.3.11	SecureScreenDockStation not secure . . . . .	26
5.3.12	Exception in updateLocation . . . . .	26
5.3.13	Buttons on CDockable . . . . .	26
<b>6</b>	<b>Version 1.0.8</b>	<b>26</b>
6.1	Incompatibilities . . . . .	27
6.1.1	DockableDisplayerListener . . . . .	27
6.1.2	EclipseThemeConnectorListener . . . . .	27
6.1.3	FocusObserver: veto . . . . .	27
6.1.4	DockProperties: priority . . . . .	27
6.1.5	CLocationModeManager . . . . .	27
6.1.6	SplitDockStation: leaf id . . . . .	28
6.1.7	DockTitleRequest . . . . .	28
6.1.8	DockFrontend: methods moved . . . . .	28
6.1.9	Placeholders . . . . .	28
6.1.10	EclipseTheme respects settings of StackDockStation . . . . .	28
6.2	API and Layout . . . . .	29
6.2.1	EclipseTheme, no-title and tabs . . . . .	29
6.2.2	CVetoFocusListener . . . . .	29
6.2.3	SingleTabDecider . . . . .	29
6.2.4	CDockable single-tab . . . . .	29
6.2.5	Tab placement . . . . .	29
6.2.6	TabLayoutManager . . . . .	29
6.2.7	CPanelPopup . . . . .	30
6.2.8	Access to actions of CDockable . . . . .	30
6.2.9	Minimum size on FlapDockStation . . . . .	30
6.2.10	CContentArea.setMinimumAreaSize . . . . .	30
6.2.11	AbstractDockable: KeyListener . . . . .	30
6.2.12	CVetoClosingListener . . . . .	30
6.2.13	AbstractCDockable: createCommonDockable . . . . .	30
6.2.14	Merger . . . . .	31
6.2.15	StackDockStation: single tab . . . . .	31
6.2.16	ScreenDockStation: fullscreen mode . . . . .	31
6.2.17	TabContentFilter . . . . .	31
6.2.18	FlapDockStation: factory . . . . .	31
6.2.19	StackDockComponent: default representation . . . . .	31
6.2.20	DropDownMenu: selection if submenu . . . . .	31
6.2.21	SplitDockFullScreenProperty . . . . .	32
6.2.22	CMaximizedLocation . . . . .	32
6.2.23	OrientationToRotationStrategy . . . . .	32

6.3	Bugfixes . . . . .	32
6.3.1	EclipseStackDockComponent: Exception . . . . .	32
6.3.2	AWT and Glass-Pane . . . . .	32
6.3.3	Check before drop . . . . .	32
6.3.4	SimpleDockAction: unbinding . . . . .	32
6.3.5	StackDockStation: reselect . . . . .	32
6.3.6	RequestDimension: constructor . . . . .	32
6.3.7	MultiDockActionSource: exception . . . . .	33
6.3.8	SplitDockStation: cursor . . . . .	33
6.3.9	FlatTab: mouse events . . . . .	33
6.3.10	StackDockStation: not removed . . . . .	33
6.3.11	FlapDockStation: exception on move . . . . .	33
6.3.12	Location of missing CDockable . . . . .	33
6.3.13	Tabs: wrong size . . . . .	33
6.3.14	Tabs: missing tabs . . . . .	33
6.3.15	Exception on normalizing . . . . .	33
6.3.16	StackDockStation: move . . . . .	33
6.3.17	Placeholders: missing . . . . .	34
6.3.18	StackDockStation: selection changes . . . . .	34
6.3.19	ScreenDockStation: exception . . . . .	34
6.3.20	StackDockStation: no event . . . . .	34
6.3.21	Tabs: strange position . . . . .	34
6.3.22	DefaultStackDockComponent: popup menus . . . . .	34
6.3.23	DockRegister: missing events . . . . .	34
6.3.24	CActionSource: missing events . . . . .	34
6.3.25	CLocationModeManager: exception . . . . .	34
6.3.26	Corner components: wrong location . . . . .	34
6.3.27	Maximized CDockable: exception . . . . .	34
6.3.28	BasicStackDockComponent: exception . . . . .	35
6.3.29	Path: wrong encoding . . . . .	35
6.3.30	EclipseMenu: update icon . . . . .	35
6.3.31	StackDockStation: move . . . . .	35
6.3.32	AbstractScreenDockWindow: wrong update . . . . .	35
6.3.33	DockTitleTab: exception . . . . .	35
6.3.34	PropertyPreference: load default . . . . .	35
6.3.35	CGrid: missing Dockables . . . . .	35
6.3.36	SplitDockStation: wrong id . . . . .	35
6.3.37	SplitDockStation: update layout . . . . .	35
6.3.38	Path: encoding . . . . .	36
6.3.39	MouseFocusObserver: request focus . . . . .	36
6.3.40	SplitDockStation: placeholders . . . . .	36
6.3.41	Dockable CStation issue . . . . .	36
6.3.42	SimpleDockAction: memory leak . . . . .	36
6.3.43	CLocation: null root . . . . .	36
6.3.44	SplitDockStation: locked size . . . . .	36
6.3.45	DockUI: stalls events . . . . .	36
6.3.46	CLayoutChangeStrategy: exception . . . . .	36
6.3.47	CDockable: base location . . . . .	36
6.3.48	Glass Extension . . . . .	37
6.3.49	SplitDockStation: cursor . . . . .	37

## Abstract

This document describes the most important changes between versions, and how developers should change their application in order to use new features. This document does not make any distinction between the core-library and the common-project. Not all changes are listed up in this document, only those enhancements which might be interesting for the majority of developers.

# 1 Version 1.0.3

Version 1.0.3 emphasizes on background enhancements. The API remains unchanged for most parts.

## 1.1 Incompatibilities

These changes break with the API from 1.0.2, clients must change their interfaces in order to work properly.

### 1.1.1 DefaultKeyboardController

**Short** The class `DefaultkeyBoardController` has been renamed to `DefaultKeyboardController`

**Reason** The new name looks better

**Clients** Replace any occurrence of `DefaultkeyBoardController` to `DefaultKeyboardController`

### 1.1.2 DefaultDockable/DefaultCDockable

**Short** `DefaultDockable` and `DefaultCDockable` now have `BorderLayout` set as default `LayoutManager`

**Reason** `BorderLayout` is the most often used `LayoutManager`.

**Clients** If another `LayoutManager` than `BorderLayout` is needed, set it up.

### 1.1.3 CDockableListener

**Short** `CDockableListener` divided into `CDockableStateListener` and `CDockablePropertyListener`

**Reason** `CDockableListener` was too big. Most clients either need information about the state, or about the properties of a `CDockable`. The case that both informations are needed is seldom.

**Clients** Need to decide which listener they implement. Note that `CDockableAdapter` implements both listeners, but not all methods get invoked when the adapter is registered only as one kind of listener.

#### 1.1.4 FlapDockStation

**Short** FlapDockStations layout is stored in a new format. The xml format will do the transition automatically, but the `DataInput/OutputStream` will not work properly.

**Reason** the old format did not carry enough information

**Clients** Store the layout in xml-format and load it again to do the transition.

#### 1.1.5 XML

**Short** `XElement` now extends `XContainer`, and no longer `XAttribute`. `XAttribute` extends `XContainer` as well.

**Reason** An element of a xml file is not an attribute, that is now reflected in the class structure

**Clients** May need to replace some occurrences of `XAttribute` by `XContainer`

#### 1.1.6 DockTheme

**Short** The common-project uses its own set of `DockThemes`. Each theme `XTheme` gets replaced by `CXTheme`

**Reason** The new themes make use of the new `ColorMap`

**Clients** Should use the new themes when possible. The old themes will work, but the user will see less features.

#### 1.1.7 DockFactory

**Short** `DockFactories` can now create any `Object` they want, and are no longer required to create `DockLayouts`. `DockLayout` has been converted into a class that wraps the `Object` that was created by a `DockFactory`

**Reason** All `DockLayouts` need to do the same things, hence clients would need to write the same code over and over again. Clients have now more freedom in how to implement `DockFactory`

**Clients** Should remove all occurrences of `implements DockLayout` and the methods `set/getFactoryId` that were defined in `DockLayout`

### 1.2 Features

This is the set of new features.

#### 1.2.1 SplitDockStation

**Short** The tree of elements of a `SplitDockStation` is now accessible from outside and can be modified directly

**Reason** It is more intuitive to work directly with the tree, some new algorithms work on the tree and are easier to implement that way.

### 1.2.2 SplitLayoutManager

**Short** New `SplitLayoutManager` calculates where to drop, and how to divide, elements of a `SplitDockStation`

**Reason** New features, like the locked size of `CDockable`, were only possible if the behavior of a `SplitDockStation` can be changed on runtime.

### 1.2.3 CDockable resize lock

**Short** The size of a `CDockable` can be locked during resize of its parent. See `setResizeLocked`, a method of `AbstractCDockable`.

**Reason** This was a request from a user

### 1.2.4 FlapLayoutManager

**Short** `FlapDockStation` now uses `FlapLayoutManager` to arrange its children

**Reason** Exchangeable behavior was a requirement for new features in the common-project.

### 1.2.5 ColorManager/ColorScheme

**Short** Many graphical elements now use `ColorManager` and `ColorSchemes`

**Reason** Colors can now be exchanged by clients. The control goes deep, even the color of a single element can be exchanged without affecting other elements of the same kind.

### 1.2.6 ColorMap

**Short** `CDockable` uses a `ColorMap` to define special colors for tabs and titles that are related to the `CDockable`

**Reason** This was a request from a user

### 1.2.7 LookAndFeel

**Short** Changes of `LookAndFeel` noted by `DockController` and forwarded to all `UIListeners`.

**Reason** Because the `ColorManager` would not be informed of the new `LookAndFeel` otherwise

### 1.2.8 CDockable resize request

**Short** `CDockables` can now request a size they would like to have, and in most environments they will get this size. See the method `setResizeRequest` of `AbstractCDockable`.

**Reason** This was a request from a user



## 2 Version 1.0.4

Version 1.0.4 introduces a few new features that add customizability

### 2.1 Incompatibilities

These changes break with the API from 1.0.3, clients must change their interfaces in order to work properly.

#### 2.1.1 Binary file format

**Short** The binary file format has been changed

**Reason** The format now includes version numbers so that backwards compatibility should be possible in the next versions

**Clients** Need to delete all binary files. They might try to write their properties with the old version in xml, and then load the xml file with the new version. This should convert the files.

#### 2.1.2 DockableListener

**Short** Has an additional method `titleExchanged`

**Reason** Allows to exchange a `DockTitle` while the `Dockable` is visible

**Clients** Need to update any class that implements `DockableListener`.

#### 2.1.3 Title visibility on CDockables

**Short** Any `CDockable` can now hide its titles at any time

**Reason** user request

**Clients** Need to update any class implementing `CDockablePropertyListener` since that listener has an additional method `titleShownChanged`.

#### 2.1.4 BasicDropDownButtonHandler

**Short** Requests now a `BasicDropDownButtonTrigger` instead of a `BasicTrigger`

**Reason** to allow steering any drop down action with the keyboard.

**Clients** unlikely to have an effect on any client

#### 2.1.5 CDockable.getClose

**Short** Method has been moved into `CommonDockable`

**Reason** The action can now be replaced through `CDockable.getAction`. There is no need for any client to replace the action by replacing the whole `DockActionSource`

**Clients** should use `putAction`, a method of `AbstractCDockable` to exchange the close-action. No fix for clients which added additional elements to the close-source.

### 2.1.6 CLocation

**Short** Additional CLocations, some methods have been moved

**Reason** To allow the new CStation more flexible CLocations were needed.

**Clients** No general solution available, clients should recompile their project and check all compiler errors.

### 2.1.7 working area

**Short** Every CStation can now be a working area

**Reason** To allow more flexibility in grouping CDockables

**Clients** That should not be visible for any client using version 1.0.3

## 2.2 Features

This is the set of new features.

### 2.2.1 Border around BubbleDisplayer

**Short** BubbleDisplayer now shows a border if the title is not null, or if the dockable is not a station

**Reason** Looks better

### 2.2.2 Backup factories (core)

**Short** DockFrontend and PredefinedDockSituation can now use backup factories. These factories are used to load elements which should be in the cache, but are missing. In case of DockFrontend they are automatically added to the frontend.

**Reason** Removes the need to add all Dockables to a DockFrontend before loading a layout from a file.

### 2.2.3 Backup factories (common)

**Short** CControl now supports lazy initialisation of SingleCDockables through the SingleCDockableBackupFactory.

**Reason** saves memory

### 2.2.4 Unregister factories from DockFrontend

**Short** DockFactorys can now be unregistered from DockFrontend

**Reason** Was missing

### 2.2.5 Action support keyboard

**Short** DockActions are triggered by pressing SPACE on the focused button, DropDownActions pop up when the DOWN (non numpad) key is pressed

**Reason** Ongoing work to allow navigating in DF without the mouse.

### 2.2.6 FocusTraversalPolicies

**Short** New FocusTraversalPolicies allow to navigate within all elements of a DockableDisplayer (including title).

**Reason** Ongoing work to allow navigating in DF without the mouse.

### 2.2.7 override predefined actions

**Short** CDockable has an additional method `getAction` which is used by various modules to override their default actions.

**Reason** Answer to a user request

### 2.2.8 CBlank

**Short** New action CBlank, which does not show anything.

**Reason** As value for `CDockable.getAction` when a predefined action should be hidden

### 2.2.9 CStation

**Short** Additional interface CStation in common. Two new stations: CMinimizeArea and CGridArea.

**Reason** Allows clients to add their own DockStations to CControl, allows to create other layouts than the "one center, four minimize areas"-layout.

## 2.3 Bugfixes

These are the bugs that were fixed/

### 2.3.1 BubbleDisplayer.getDockableInsets

**Short** The method did not calculate its result correctly.

**Reason** A flaw in the design of BasicDockableDisplayer

### 2.3.2 IndexOutOfBoundsException from ButtonPanel

**Short** The exception was thrown when an invisible action was on the panel

**Reason** invisible actions were not considered when writing ButtonPanel

### 2.3.3 Mode change of CDockable

**Short** CDockable did not go into normalized-mode when externalized and never normalized before

**Reason** Properties were missing and could not be created automatically

### 2.3.4 Opening maximized CDockable

**Short** CDockable could not be opened maximized.

**Reason** framework got confused because CDockable did not have a parent.

### 2.3.5 Unbind of DockAction called to often

**Short** A DockAction could throw an exception "unbind called to often"

**Reason** When a DockAction was a child of a MenuMenuHandler, its unbind method was called even if the action was not displayed. However the bind action was called only if the action was displayed, so the internal counter was no longer correct. Every time a menu with such an action was shown, the counter was decremented by one. When it reached a value below 0, an exception was thrown. Since an action could be bound by many elements, the exception occurred at random places.

## 3 Version 1.0.5

Version 1.0.5 brings the possibility to navigate around only by hitting some keys on the keyboard. When clicking the `ctrl+shift+e` combination, a dialog opens on which a Dockable can be selected.

DockActions in button form can be activated with `space`, and the dropdown actions menu can be opened with the `arrow down` key.

This release contains some tricky incompatibilities which need to be handled very carefully.

### 3.1 Incompatibilities

The changes that need special care.

#### 3.1.1 DockStationListener

**Short** The method `dockableSelected` of `DockStationListener` has an additional parameter that indicates which element was selected before the change.

**Reason** No need for listeners to store the old values.

**Clients** Must carefully update all classes and interfaces that implement `DockStationListener`. Be especially careful not to mix up the new arguments with the old ones.

### 3.1.2 DockableFocusListener

**Short** The `DockableFocusListener` has been divided into two interfaces: `DockableFocusListener` and `DockableSelectionListener`. The remaining method in `DockableFocusListener` now takes a `DockableFocusEvent` and no longer directly the involved elements. The class `DockableFocusAdapter` has been deleted.

**Reason** Events allow further changes of the system without change of the `DockableFocusListener` itself. Since every client needs to update its methods anyway, `DockableFocusAdapter` could be deleted.

**Clients** Should use `DockableFocusListener` instead of `DockableFocusAdapter`.

### 3.1.3 DockTheme.getDockableSelection

**Short** `DockTheme` has an additional method `getDockableSelection`.

**Reason** A `DockableSelection` is needed to change the focused `Dockable` using only the keyboard. Since `DockableSelection` is a graphical element, it has to be handled by the `DockTheme`.

**Clients** Should implement the missing method in their `DockThemes`. Using `DefaultDockableSelection` is an easy solution.

### 3.1.4 tap-strip no longer painted by TapPainter

**Short** `TabPainter` does no longer paint the tab-strip directly. It now creates a `TabStripPainter` that paints the strip.

**Reason** The new object can work with the color map.

**Clients** Have to provide a `TabStripPainter` as well.

### 3.1.5 KeyboardController does fire less events

**Short** The `KeyboardController` does no longer fire events when it could not find the source-`Dockable` of the event. As a result the `KeyboardListener` does no longer receive `null` as argument of any of its methods.

**Reason** Events were fired which had nothing to do with the framework at all.

**Clients** If they need all key events, then they can add a global `KeyListener` to `KeyboardController` using the method `addGlobalListener`.

### 3.1.6 ComponentHierarchyObserver

**Short** The `ComponentHierarchyObserver` includes more `Components` in its search. The `ComponentHierarchyObserverListener` now works with an event and does no longer receive all the elements as arguments.

**Reason** Allows more features to work correctly in restricted environments.

**Clients** Need to be aware that not every `Component` that is found by the observer is a child of a `Dockable`.

## 3.2 API and Layout

A list of new API elements and changes that affect the layout.

### 3.2.1 KeyStroke for closing Dockable

**Short** The `KeyStroke` for closing a `CDockable` or `Dockable` has been changed from `ctrl+c` to `ctrl+F4`.

**Reason** Andrew pointed out, that `ctrl+c` is already used by many applications...

### 3.2.2 New listeners

**Short** There are new listeners, `CFocusListener`, `CKeyListener` and `CDoubleClickListener`, which can be added to `CDockable` or to `CControl` if all `CDockables` should be monitored.

**Reason** Might be helpful for some applications

### 3.2.3 ComponentHierarchObserver

**Short** Clients can now add and remove `Components` from the `ComponentHierarchyObserver`. The observer also includes `DockTitles` in its search for `Components`.

**Reason** Might become necessary for complex applications that run in a restricted environment.

### 3.2.4 Root window for DockController

**Short** The `DockController` can now find the root window of the application. The window can also be set directly using `setRootWindow`. If so, then the root window is added to the `ComponentHierarchyObserver`.

**Reason** Necessary to show small dialogs like the new `DockableSelector`

### 3.2.5 FocusTraversalPolicies

**Short** All `DockThemes` now support `FocusTraversalPolicies`. Now each `DockAction` and all `Components` of a `Dockable` can be reached by using only the keyboard.

**Reason** A nice feature for people which do not like the mouse

### 3.2.6 Dialog to select focused Dockable

**Short** The `DockableSelector` and `DockableSelection` allow users to select the focused `Dockable` using only the keyboard. The feature is activated as soon as `ctrl+shift+e` is pressed.

**Reason** A nice feature for people which do not like the mouse

### 3.2.7 Extracting colors from LookAndFeel

**Short** The mechanism to read colors from `LookAndFeel`s has been upgraded. Each `LookAndFeel` can now have its own specialized `LookAndFeelColors` that reads the colors.

**Reason** Allows to be more flexible with colors, allows the correct use of Nimbus and Windows.

### 3.2.8 EclipseTheme

**Short** `EclipseTheme` uses more colors from the `LookAndFeel`

**Reason** looks better

### 3.2.9 SplitDockStation

**Short** When dropping an element onto a `SplitDockStation`, the elements that are put aside receive at least a quarter of their original size.

**Reason** Sometimes the old elements shrunk too much.

## 3.3 Bugfixes

### 3.3.1 Missing colors for BasicTheme

**Short** `BasicTheme` did not update colors for the keys `paint.line`, `paint.divider` and `paint.division`. As a result some painting was not as in the older versions.

### 3.3.2 Cutting bounds of children of SplitDockStation

**Short** The bounds of children of `SplitDockStation` are now cut such that they are always within the stations boundaries.

**Reason** Rounding errors sometimes lead to little failures that made a single line of pixels invisible.

### 3.3.3 NullPointerException when changing focus

**Short** A `NullPointerException` could be thrown when the focus changed.

### 3.3.4 Undecorated dialogs not undecorated

**Short** When using `LookAndFeel`s that can draw window decorations on their own (like `JTattoo`), then `FlapWindow`, `ScreenDockDialog` and others could have decorations.

**Reason** The flag that advices the `LookAndFeel` not to paint a decoration was not set in the `JRootPanels` of these windows.

### 3.3.5 `RexTabbedComponent` not adding/removing children

**Short** `RexTabbedComponent` does no longer add and remove its children to change their visibility, it now uses a `CardLayout`.

**Reason** Some `Components` did miss the change of the `LookAndFeel` when they were a child of `RexTabbedComponent`.

### 3.3.6 Focusing a hidden `CDockable`

**Short** When focusing a normalized `CDockable` that was hidden behind a maximized `CDockable`, then the focused dockable did not became visible.

**Reason** An old security system prevents change of the maximized element by the focus system.

### 3.3.7 Missing events when changing state of `CDockable`

**Short** When the `ExtendedMode` of a `CDockable` did not change because of a call of a special method, no state-change-events were fired.

**Reason** It was not intended that one action could change the state of many `CDockables`.

## 4 Version 1.0.6

This version brings the preference system. The API was changed at some places in order to bring the preference system to work.

### 4.1 Incompatibilities

The changes that need special care.

#### 4.1.1 Dockable with Tooltip

**Short** `Dockable` has a new method `getTitleToolTip`. `DockableListener` has a new method `titleToolTipChanged`.

**Reason** Allows to show a tooltip for a `Dockable` on titles and on tabs.

**Clients** Must implement the two new methods.

#### 4.1.2 `ColorManager` generalized

**Short** `ColorManager` extends `UIProperties`, `ColorProvider` is replaced by `ColorBridge` which extends `UIBridge`, `DockColor` extends `UIValue`. `ColorManager.getProviderFor` is replaced by `UIProperties.getBridgeFor`. Bridges and Values are no longer connected though the class of the `UIValue` but by a `Path` object. These objects are much more flexibel than classes and not hard to understand.



**Reason** This generalization will allow to use the `UIProperties` for other things than just colors. There are plans to use the same system for fonts as well.

**Clients** Should replace `ColorProvider` by `UIBridge`

#### 4.1.3 Resize Request in Common

**Short** Size requests are now handled by `RequestDimension` and no longer with `Dimension`.

**Reason** Allows to issue requests only for width or for height.

**Clients** Have to replace occurrences of `Dimension` by `RequestDimension`.

#### 4.1.4 DockElementRepresentative

**Short** `Dockable` and `DockTitle` implement the interface `DockElementRepresentative`

**Reason** Allows unified access to all `Components` which are linked to a `Dockable`.

**Clients** Have to implement the additional methods of `DockElementRepresentative`

#### 4.1.5 SimpleModifierMask deleted

**Short** The class `SimpleModifierMask` has been removed. The interface `ModifierMask` has been changed to be a class effectively replacing `SimpleModifierMask`.

**Reason** This was necessary for the preference system. It was also unlikely that a client would ever implement `ModifierMask`.

**Clients** Must replace `SimpleModifierMask` by `ModifierMask`.

#### 4.1.6 Map of DockThemes

**Short** `CControl` has now a `ThemeMap`. This map contains `String-ThemeFactory` pairs. A new theme can be activated by calling `ThemeMap.select`.

**Reason** This is a simple representation of all the choices a user can do. The `CThemeMenuPiece` and the preference system can use the map to show choices and selection.

**Clients** Instead of using `CControl.setTheme( DockTheme )` they should use `CControl.setTheme( String )`. Additional `ThemeFactory`s have to be added directly to the `ThemeMap`, `CThemeMenuPiece` does no longer support inserting factories.

#### 4.1.7 Persistent storage of DockTheme

**Short** The DockTheme of a CControl is no longer stored by the CThemeMenuPiece but directly by its ThemeMap.

**Reason** The ThemeMap is always present, the CThemeMenuPiece not. Hence if the ThemeMap is responsible for storing the theme, then the theme gets always stored.

**Clients** Cannot do anything. The setting of the theme will be lost the next time the application starts and has to be set anew.

## 4.2 API and Layout

A list of new API elements and changes that affect the layout.

### 4.2.1 Dropping onto SplitDockStation

**Short** When dropping something onto a SplitDockStation, the old content always gets at least 25% of the remaining space.

**Reason** In some situations the old content get no space and became invisible.

### 4.2.2 UIProperties

**Short** New UIProperties, a generalisation of ColorManager.

**Reason** Precondition to implement a similar system for fonts.

### 4.2.3 Opened LockedResizeLayoutManager

**Short** The private inner classes of LockedResizeLayoutManager have been made public and top level.

**Reason** Clients have better access and can better customize LockedResizeLayoutManager.

### 4.2.4 ConflictResolver for locked resize

**Short** The ConflictResolver in Common can now be used to resolve conflicts on resize when locked CDockables are around. Can be applied using the key CControl.RESIZE\_LOCK\_CONFLICT\_RESOLVER.

**Reason** Developers wished to have the choice between different behaviors.

### 4.2.5 FullLockConflictResolver

**Short** A new ConflictResolver which is inspired by the behavior of VLDocking

**Reason** User request

#### 4.2.6 DockElementRepresentative

**Short** New interface `DockElementRepresentative`. Creates a link between a `Component` and a `DockElement`.

**Reason** Gives a unified way to handle popup menus and drag and drop operations.

#### 4.2.7 Common: close-action and setVisible

**Short** Clicking onto the close-action and calling `setVisible( false )` on a `CDockable` will now have the exact same effects.

**Reason** Seems to be reasonable that the close action just calls `setVisible`.

#### 4.2.8 Preference system

**Short** A new system has been put in place to handle preferences. This new system is located in the package `bibliothek.extension.gui.dock`.

**Reason** This new system allows users to see and change various properties of the library. This includes things like the shortcuts for actions (like `ctrl+m` for maximizing a `Dockable`) or which colors are used by `BubbleTheme`. Future releases might contain more preferences.

#### 4.2.9 ColorScheme as property

**Short** `BasicTheme` and subclasses read their `ColorScheme` from the `DockProperties`.

**Reason** a condition for the preference system

#### 4.2.10 Default locations in Common

**Short** Clients can set the default location of a `Dockable` in `Common`. The method `setLocation` of `CStateManager` can be used for that. Also `AbstractCDockable` has a new method `setDefaultLocation` which can be used even if the element is not yet added to a `CControl`.

**Reason** user request.

#### 4.2.11 Borders on OverpaintablePanel

**Short** `OverpaintablePanel` now supports `Borders`.

**Reason** Every `Component` should support `Borders`.

#### 4.2.12 SplitDockStation can disabled resizing

**Short** Resizing on a `SplitDockStation` can be disabled.

**Reason** Requested by a user.

#### 4.2.13 Handle AWT components

**Short** The `AWTComponentCaptureStrategy` can be used to create images from AWT components.

**Reason** AWT components cannot be handled like Swing components, the mechanism normally used created just a blank image.

### 4.3 Bugfixes

#### 4.3.1 `DefaultConflictResolver` did not respect locked sizes

**Short** When several `ResizeRequests` with different priority had to be handled, `DefaultConflictResolver` did not respect all of them. The algorithm has been fixed.

#### 4.3.2 Opening maximized `CDockable`

**Short** When opening a `CDockable` which would stack on a maximized `CDockable`, then the layout could get scrambled. The solution is now to unmaximize any `CDockable`, then add the new element, then re-maximize the `CDockables`.

#### 4.3.3 Dropping Dockable on `SplitDockStation`

**Short** Dockables can now be dropped onto `SplitDockStations` which have size 0/0. In earlier versions the divider between `Dockables` had a fixed size in pixels. Now the size of the divider is set to 0 if the `SplitDockStation` is too small. This prevents children to have negative sizes.

#### 4.3.4 `CSplitLocation` broken

**Short** `CSplitLocation.expandProperty` did process the first element of a tree-path twice (thanks srcnick for fixing this bug).

#### 4.3.5 `CStateManager.getLocation` broken

**Short** `CStateManager.getLocation` did return null when it should produce a result. There were also some `CLocations` which did not return the correct result causing `getLocation` to fail.

#### 4.3.6 Stack-component of `EclipseTheme` broken

**Short** When removing all elements of `EclipseStackDockComponent`, some elements could remain invisible.

#### 4.3.7 Change `ColorScheme` could throw NPE

**Short** When updating the colors of a `BasicDockTheme` which was not installed, a `NullPointerException` was thrown.

#### 4.3.8 Items in popup-menu did do nothing

**Short** Some `DockActions` were not correctly wired when in a popup-menu. Clicking them would not result in any action (affects all `SelectableDockActions`).

## 5 Version 1.0.7

Version 1.0.7 emphasizes on details. Many bugfixes are included and new settings allow further customization. The layout-storage mechanism has been improved to support missing dockables.

### 5.1 Incompatibilities

The changes that need special care.

#### 5.1.1 `DockableDisplayerHints`

**Short** `Dockable` has a new method `configureDisplayerHints`.

**Reason** This allows `Dockables` to communicate with their `DockableDisplayers`. For example a `SplitDockStation` tells its displayer to paint a border if the station has no children, but not to paint a border if it has children.

**Clients** If implementing `Dockable` directly need to add this method.

#### 5.1.2 `ScreenDockDialog` extends new class

**Short** `ScreenDockDialog` extends other classes than before.

**Reason** The whole management of dialogs for `ScreenDockStation` has been rewritten. A `ScreenDockStation` now supports any kind of window, not only dialogs.

**Clients** Should not be affected.

#### 5.1.3 `DockFactory` uses a new layer

**Short** The whole layout-storage mechanism has been updated.

**Reason** To support missing dockables a new layer containing meta information was necessary.

**Clients** The interface `DockFactory` contains new and changed methods

#### 5.1.4 `CGridArea` implements `CDockable`

**Short** `CGridArea` is no longer just a panel but can also be used as dockable.

**Reason** In order to use `CGridArea` as superclass for `CWorkingArea` this interface was needed.

**Clients** Should not affect clients.

### 5.1.5 CWorkingArea extends CGridArea

**Short** CWorkingArea is now a subclass of CGridArea.

**Reason** CWorkingArea and CGridArea have almost the same behavior. New interfaces and lesser coupling allowed to reuse CGridArea. In a future release they might even be merged into one class.

**Clients** Most clients should not notify this change, clients that use code like `x instanceof CGridArea` need to be updated.

### 5.1.6 CControlFactory: no longer creates CWorkingAreas

**Short** CControlFactory no longer creates CWorkingAreas but SplitDockStations.

**Reason** This is part of an ongoing effort to lessen the coupling of classes in Common.

**Clients** Clients that used a customized CControlFactory may need to update their factory.

### 5.1.7 CommonDockable: getClose replaced with getSources

**Short** In CommonDockable the method getClose was replaced with getSources.

**Reason** This way a CommonDockable can support more than just one special DockActionSource. Also the close-action-source is no longer integrated that tight into the system.

**Clients** Since clients should not work on this level anyway they don't need to worry.

### 5.1.8 PropertyKey: requires factory for default value

**Short** PropertyKey requires a PropertyFactory to set up its default value.

**Reason** Fixes a memory leak by preventing PropertyKey from sneaking in global variables.

**Clients** Need to implement the factory if they create new PropertyKeys.

## 5.2 API and Layout

A list of new API elements and changes that affect the layout.

### 5.2.1 Button-title supports colors

**Short** The button-title on FlapDockStation can change its color, new keys for that feature are provided in ColorMap.

**Reason** Was just missing.

### 5.2.2 FlapDockProperty: support state and size

**Short** FlapDockProperty stores now holding state and window size as well.

**Reason** Allows clients more control over the layout, was necessary for Common.

### 5.2.3 AdjacentDockFactory

**Short** The AdjacentDockFactory can store additional information about a Dockable when writing a layout.

**Reason** Was required for Common.

### 5.2.4 DockSituation: support for missing elements

**Short** DockSituation has a number of new methods to support missing or invisible elements. The new methods are fillMissing and estimateLocations.

**Reason** Makes the user interface more consistent if missing elements are made available later.

### 5.2.5 Support for gaps in layout

**Short** DockFrontend now tries to fill missing gaps in the layout before a layout is applied.

**Reason** Makes the user interface more consistent if missing elements are made available later.

### 5.2.6 Storing information of invisible dockables

**Short** DockFrontend stores layout information of invisible/missing elements.

**Reason** Otherwise information would be lost and the user interface would seem inconsistent.

### 5.2.7 Access to information of missing dockable

**Short** New methods listFrontendEntries and getFrontendEntry in DockFrontend.

**Reason** The methods allow access to all information of missing (and normal) Dockables.

### 5.2.8 PreferenceTable: order of operations reversed

**Short** The order of “default” and “remove” operation are reversed.

**Reason** The “default” operation should be the last operation.

### 5.2.9 Automatic stack creation in CGrid

**Short** Eduardo Born suggested that if some dockables are placed at the same location in a CGrid then they should be put together in a stack.

**Reason** There is no reason not to do it this way.

### 5.2.10 Central register for CDockables

**Short** CControl stores all its stations and dockables now in a CControlRegister.

**Reason** New classes are introduced to free CControl of minor tasks.

### 5.2.11 FontManager

**Short** New methods to change the fonts on titles and tabs. The interface DockFont provides some keys that can be used together with FontManager.

**Reason** That's a feature every docking-framework should have

### 5.2.12 FontMap

**Short** Common supports the new font system, the FontMap can be used like the ColorMap.

**Reason** Because Core allows this.

### 5.2.13 More than one maximize-area

**Short** In Common more than only one station can now be marked as being potential parent of a maximized CDockable.

**Reason** Part of ongoing work for less coupling in Common.

### 5.2.14 Veto before changing layout

**Short** The method hiding of VetoableDockFrontendListener is called when setting a new layout.

**Reason** Prevents dockables to disappear that must always be visible

### 5.2.15 CGrid/SplitDockGrid: preselect element

**Short** CGrid/SplitDockGrid have a new method select/setSelected to select a CDockable/Dockable in a stack of dockables.

**Reason** More control over the layout.



### 5.2.16 WindowProviders

**Short** New interface `WindowProvider` allows to change the root-window even after the framework runs.

**Reason** Some clients do not know their root window when setting up a controller, other clients did have a hard time to find the root-`JFrame` (like applets, which do not have such a frame).

### 5.2.17 AppletWindowProvider

**Short** A new `WindowProvider` is available, the `AppletWindowProvider`.

**Reason** This class supports `Applets` by finding the (normally hidden) window on which the applet lies.

### 5.2.18 LocaleListener

**Short** The new `LocaleListener` can be added to `DockUI` and will be informed if the `Locale` changes.

**Reason** Internal caches of `Common` can be cleaned through this listener.

### 5.2.19 DockController: freeze layout

**Short** `DockController` has new methods `freezeLayout`, `meltLayout` and `isLayoutFrozen`.

**Reason** These methods temporarily freeze the layout so clients can safely add and remove `Dockables` from the tree. Prevents the `SingleParentRemover` to do its work and change the tree at the same time.

## 5.3 Bugfixes

### 5.3.1 SplitDockStation not respecting acceptances

**Short** On a `SplitDockStation` a `Dockable` could be dropped over another element which didn't accept that combination.

### 5.3.2 Common and CGrid: not supporting big stacks

**Short** When dropping `CGrid` with stacks that have 3 or more elements, then an exception was thrown.

### 5.3.3 DockFrontend did not read setting correctly

**Short** The method `read` of `DockFrontend` did mark the main setting as simple entry while it should have been marked as main entry.

### 5.3.4 Infinite recursion in focus reversal

**Short** `DockFocusTraversalPolicy` would create an infinite recursion when used together with `javax.swing.LegacyGlueFocusTraversalPolicy`.

### 5.3.5 CWorkingArea not settings itself as working-area

**Short** The method `deploy` of `CWorkingArea` did not inform the children that they are now child of a working-area.

### 5.3.6 SplitDockGrid throwing exception

**Short** When the same coordinates were used twice or more for putting elements in a `SplitDockGrid` an exception was thrown.

### 5.3.7 FlapWindow not resizing

**Short** A `FlapWindow` did not always resize correctly when its parent got resized while the window was invisible. Fixed by Peter.

### 5.3.8 Dropping CWorkingArea

**Short** Dropping a `CWorkingArea` that has children did not work.

### 5.3.9 Drop CDockable with no location but working-area

**Short** A `CDockable` that has not set any location but belongs to a working-area will now use the default location for that working-area.

### 5.3.10 AbstractCDockable ignores settings

**Short** `AbstractCDockable` did not respond when setting an extended mode and another extended mode was disabled. The cause of this failure was a missing “break” in a “switch” statement.

### 5.3.11 SecureScreenDockStation not secure

**Short** `SecureScreenDockStation` was not using `SecureScreenDockWindowFactory`.

### 5.3.12 Exception in updateLocation

**Short** The method `updateLocation` of `DockFrontend` would throw an exception if a `Dockable` in the tree was at the same time a root-station.

### 5.3.13 Buttons on CDockable

**Short** When a minimized `CDockable` was closed and then made visible again its `extension-mode-buttons` were not correctly set.

## 6 Version 1.0.8

This version brings many new features: the `CLocationModeManager` handles the location of `CDockables`, placeholders store the location of `Dockables` much more precise than the old system, tabs can be placed at any side.

The changes are ordered by the time when they were introduced.

## 6.1 Incompatibilities

The changes that need special care.

### 6.1.1 DockableDisplayerListener

**Short** `DockableDisplayers` can now be observed by `DockableDisplayerListeners`.

**Reason** This allows a `DockableDisplayer` to mark itself as obsolete, `DockStations` then can create new displayers.

**Clients** Any class that uses `DockableDisplayers` should add the listener and react on its events.

### 6.1.2 EclipseThemeConnectorListener

**Short** `EclipseThemeConnector` can be observed by a `EclipseThemeConnectorListener`.

**Reason** This allows to change properties even after asking the connector for them

**Clients** Clients implementing a `EclipseThemeConnector` have to call the listener

### 6.1.3 FocusObserver: veto

**Short** Focus transfer when clicking the mouse can now be canceled. The `MouseFocusObserver` calls a method `handleVeto` which cancels an event.

**Reason** A feature request by a user, should be used with care.

**Clients** The new `FocusVetoListener` can be added to the focus observer and allows clients to cancel focus transfer.

### 6.1.4 DockProperties: priority

! API: `DockProperties` now supports different priorities for values, "default", "theme" and "client".

**Short** `DockProperties` now stores properties with different priorities. "Client" overrides "theme", "theme" overrides "default".

**Reason** Allows `DockThemes` and clients to use the properties together.

**Clients** Should always register their properties with priority "client".

### 6.1.5 CLocationModeManager

**Short** `CStateManager` got replaced by `CLocationModeManager`.

**Reason** The `CLocationModeManager` is built much more generic than `CStateManager` allowing new `DockStations` and new extended-modes.

**Clients** The method `CControl.getStateManager` was renamed to `getLocationManager`.

### 6.1.6 SplitDockStation: leaf id

**Short** Each node of a `SplitDockStation` has now a unique identifier.

**Reason** This identifier is used to store the location of a `Dockable`.

**Clients** This is a change in the internal API, clients are not affected.

### 6.1.7 DockTitleRequest

**Short** `DockTitleFactory` has been changed. Now a `DockTitleRequest` is installed on the factory, the factory may trigger this request anytimes (also more than once). The new class `StationChildHandle` manages the interaction between `Dockable`, `DockableDisplayer`, `DockTitle` and `DockTitleRequest`.

**Reason** The API is now much more consistent. Also anyone knowing a `DockTitleRequest` can update/replace the title at any time.

**Clients** Please read chapter 5 “Titles” of the guide for Core.

### 6.1.8 DockFrontend: methods moved

**Short** Some methods of `DockFrontend` have been moved to `DefaultLayoutChangeStrategy`.

**Reason** Allows `Common` to handle its special needs, e.g. to reuse `MultipleCDockables` when the layout changes.

**Clients** This API was used internally, clients should not be affected.

### 6.1.9 Placeholders

**Short** Placeholders remain at the position where a `Dockable` was removed. The feature is implemented in `Core` but only activated if `Common` is used.

**Reason** This makes location information much more resilient against failures due to missing `Dockables`.

**Clients** May need to implement a `PlaceholderStrategy` if they want to use placeholders in `Core`. All existing layout information will be upgraded automatically. It is not possible for an old version of `DockingFrames` to read a layout written with 1.0.8.

### 6.1.10 EclipseTheme respects settings of StackDockStation

**Short** `BaseTabComponent` is no longer responsible for keeping icon and text up to date, this must be handled by the `StackDockComponent`.

**Reason** This means that the `EclipseTheme` now respects the text and icon set by the `StackDockStation`

**Clients** This API was used internally, clients should not be affected by these changes

## 6.2 API and Layout

A list of new API elements and changes that affect the layout.

### 6.2.1 EclipseTheme, no-title and tabs

**Short** New `CommonEclipseThemeConnector` ensures that `CDockable` with no title do not have a tab if not necessary in the `EclipseTheme`.

**Reason** Since tabs are used like titles in the `EclipseTheme`, they also should be hidden if the no-title property is set.

- API: new `CVetoFocusListener` in `Common`, can speak a veto for some focus changes

### 6.2.2 CVetoFocusListener

**Short** A new `CVetoFocusListener` can be added to `CControl`.

**Reason** The listener can be used to cancel focus changes, should be used with care.

### 6.2.3 SingleTabDecider

**Short** The interface `SingleTabDecider` allows `Dockables` to be displayed with a tab, even if a tab is not necessary.

**Reason** Creates a new look and a new feeling if combined with a no-title feature.

### 6.2.4 CDockable single-tab

**Short** `CDockable` has a new property `singleTabShown` which is forwarded to a `SingleTabDecider`

**Reason** Allows clients to enable/disable single tabs for each `CDockable` individually.

### 6.2.5 Tab placement

**Short** New property `StackDockStation.TAB_PLACEMENT` globally sets where to put tabs. Must be supported by the current `TabLayoutManager`.

**Reason** Allows clients more customization.

### 6.2.6 TabLayoutManager

**Short** New class `TabLayoutManager` to handle the positioning of tabs on a `StackDockStation`.

**Reason** Decoupling of looks (tabs) and logic (their position).

### 6.2.7 CPanelPopup

**Short** New action CPanelPopup in Common provides an easy way to create a popup-action with an arbitrary Component as content.

**Reason** Allows more customization.

### 6.2.8 Access to actions of CDockable

**Short** CDockable has new methods getAction and getActionCount to access CActions that were added.

**Reason** Clients no longer have to remember what actions they added.

- API:

### 6.2.9 Minimum size on FlapDockStation

**Short** New key FlapDockStation.MINIMUM\_SIZE sets the default minimum size of FlapDockStations.

**Reason** To make sure a Dockable does not get too small.

### 6.2.10 CContentArea.setMinimumAreaSize

**Short** CContentArea.setMinimumAreaSize set the minimum size of minimized CDockables.

**Reason** To make sure a CDockable does not get too small.

### 6.2.11 AbstractDockable: KeyListener

**Short** AbstractDockable has new method add/removeKeyListener.

**Reason** Allows to observe any KeyEvent that is related to a Dockable. Children of dockable DockStations are ignored.

### 6.2.12 CVetoClosingListener

**Short** A CVetoClosingListener can be added to CControl or CDockable.

**Reason** The listener is called before a Dockable or a set of Dockables is closed. The listener may cancel the operation.

### 6.2.13 AbstractCDockable: createCommonDockable

**Short** AbstractCDockable has a new method createCommonDockable.

**Reason** Clients can override the method and thus customize the Dockable which represents the CDockable.

#### 6.2.14 Merger

**Short** The interface `Merger` allows two `DockStations` to be merged automatically.

**Reason** The implementation `StackMerger` merges to `StackDockStations`. This allows to drag a stack of `Dockables` at a new location.

#### 6.2.15 StackDockStation: single tab

**Short** A `StackDockStation` with only one child can still show a tab depending on the type of `StackDockComponent`

**Reason** A feature request by a developer.

- API: `ScreenDockStation` now supports fullscreen mode for its children. What "fullscreen" means can be influenced by a `ScreenDockFullscreenStrategy`.

#### 6.2.16 ScreenDockStation: fullscreen mode

**Short** `ScreenDockStation` now supports fullscreen mode for its children. The exact meaning of "fullscreen" is defined by a `ScreenDockFullscreenStrategy`.

**Reason** Because it was really missing.

#### 6.2.17 TabContentFilter

**Short** The new interface `TabContentFilter` tells the framework what to show on a tab.

**Reason** Allows some more customization, e.g. show only short titles.

#### 6.2.18 FlapDockStation: factory

**Short** `FlapDockStation` supports factory for creating the window

**Reason** To use `FlapDockStation` on a `JInternalFrame`.

#### 6.2.19 StackDockComponent: default representation

**Short** `StackDockComponent` now offers a method `createDefaultRepresentation`.

**Reason** This allows code to be executed when clicking on, or dragging of, an empty space arounds tabs.

#### 6.2.20 DropDownMenu: selection if submenu

**Short** Clicking on a child action of a `DropDownAction` that is shown as submenu now changes the selection of the button as well.

**Reason** That is the expected behavior.

### 6.2.21 SplitDockFullScreenProperty

**Short** New property `SplitDockFullScreenProperty`.

**Reason** This property points to a fullscreen `Dockable` on a `SplitDockStation` and can be used to merge that element with another `Dockable`.

### 6.2.22 CMaximizedLocation

**Short** Extends now `AbstractStackholdingLocation`.

**Reason** Allows to combine a maximized `Dockable` with another `Dockable`

### 6.2.23 OrientationToRotationStrategy

**Short** New interface `OrientationToRotationStrategy`.

**Reason** The strategy allows clients to specify how to rotate text on a `DockTitle`

## 6.3 Bugfixes

### 6.3.1 EclipseStackDockComponent: Exception

**Short** `EclipseStackDockComponent` contained an unnecessary and not correctly updated list of `Dockables` causing an `IndexOutOfBoundsException` on JREs of version 1.5.0.12 or higher.

### 6.3.2 AWT and Glass-Pane

**Short** Workaround for bug [http://bugs.sun.com/bugdatabase/view\\_bug.do?bug\\_id=6797587](http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6797587) , AWT components not painted properly if behind a glass-pane.

### 6.3.3 Check before drop

**Short** `DockStations` check whether dropping some `Dockable` would create an invalid `Component-tree`, and forbid such drag and drop operations

### 6.3.4 SimpleDockAction: unbinding

**Short** `SimpleDockAction` did not unbind itself correctly due of events fired in an unfortunate order. Repaired thanks to an anonymous user.

### 6.3.5 StackDockStation: reselect

**Short** When making a selected `Dockable` on a `StackDockStation` invisible, a new `Dockable` should be properly selected.

### 6.3.6 RequestDimension: constructor

**Short** `RequestDimension( int, boolean )` did not set the height property.



### **6.3.7 MultiDockActionSource: exception**

**Short** Wrong index in `MultiDockActionSource.add( DockActionSource )` could lead to `IndexOutOfBoundsException`.

### **6.3.8 SplitDockStation: cursor**

**Short** Cursor on the `SplitDockStation` sometimes does not change from arrow to default cursor. Should happen less often now, patch by Eduardo Born.

### **6.3.9 FlatTab: mouse events**

**Short** `FlatTab` did not react on mouse-dragged events if in a "secure environment".

### **6.3.10 StackDockStation: not removed**

**Short** `StackDockStation` was not always removed if it had only one child and this one child was a `CDockable` associated with a `CWorkingArea`.

### **6.3.11 FlapDockStation: exception on move**

**Short** Moving a `Dockable` on a `FlapDockStation` did throw an exception if the destination index was too big.

### **6.3.12 Location of missing CDockable**

**Short** If a `CDockable` is removed and the `CMissingDockableStrategy` tells to store information about that `Dockable`, then its location remains now stored.

### **6.3.13 Tabs: wrong size**

**Short** If `TabPlacement` was "left" or "right", then new tabs in a `StackDockStation` could be too big. Reason for this bug was that new tabs were not properly initialized and assumed that `TabPlacement` was "top".

### **6.3.14 Tabs: missing tabs**

**Short** `StackDockStation` did not add its children correctly to the `StackDockComponent`, leading to some missing tabs

### **6.3.15 Exception on normalizing**

**Short** Normalizing a externalized `CDockable` could throw an `Exception`

### **6.3.16 StackDockStation: move**

**Short** `StackDockStation` could not be moved around in `Common` even if children would allow it.

### **6.3.17 Placeholders: missing**

**Short** Dragging a station away from another station did not store placeholders

### **6.3.18 StackDockStation: selection changes**

**Short** During drag and drop, dragging the mouse over a tab of a `StackDockStation` could exchange the selected `Dockable`.

### **6.3.19 ScreenDockStation: exception**

**Short** Dragging a `Dockable` over a child of a `ScreenDockStation` could throw an exception (in `Common`).

### **6.3.20 StackDockStation: no event**

**Short** `StackDockStation` did not fire `dockable-selected` event properly when using the `DefaultStackDockComponent`.

### **6.3.21 Tabs: strange position**

**Short** Tabs with unequal height were positioned strangely.

### **6.3.22 DefaultStackDockComponent: popup menus**

**Short** Popup menus are now enabled for tabs on a `DefaultStackDockComponent`

### **6.3.23 DockRegister: missing events**

**Short** If `DockRegister` was stalled and a `DockStation` was added/removed to the register and later `Dockables` added/removed from that station, then the `DockRegister` could miss these modifications and store too many/few `Dockables`.

### **6.3.24 CActionSource: missing events**

**Short** `CActionSource` did not fire events when removing or replacing actions

### **6.3.25 CLocationModeManager: exception**

**Short** `CLocationModeManager.setLocation` did not compare the correct objects and has always thrown an exception.

### **6.3.26 Corner components: wrong location**

**Short** Minimized `CDockables` did not appear at the correct location if corner components were used

### **6.3.27 Maximized CDockable: exception**

**Short** Dragging a maximized `CDockable` could throw an exception

### **6.3.28 BasicStackDockComponent: exception**

**Short** Dragging a tab from a `BasicStackDockComponent` could throw an exception

### **6.3.29 Path: wrong encoding**

**Short** Path did not encode non-java identifiers right.

### **6.3.30 EclipseMenu: update icon**

**Short** `EclipseMenu` did not update its icon automatically

### **6.3.31 StackDockStation: move**

**Short** `StackDockStation.move(Dockable, DockableProperty)` could throw exception if destination index was too big.

### **6.3.32 AbstractScreenDockWindow: wrong update**

**Short** `AbstractScreenDockWindow.setDockable` did call `updateTitleIcon` instead of `updateTitleText`.

### **6.3.33 DockTitleTab: exception**

**Short** `DockTitleTab` caused exception when uninstalled

### **6.3.34 PropertyPreference: load default**

**Short** Instead of doing nothing, `PropertyPreferences` now load their default value if they cannot read any other value.

### **6.3.35 CGrid: missing Dockables**

**Short** When deploying a `CGrid` some `Dockables` did not appear: the tree of the `SplitDockStation` was cleaned up too early before all `Dockables` had been inserted, leading to a corrupted tree.

### **6.3.36 SplitDockStation: wrong id**

**Short** `SplitDockStation` did assign `leaf-id` to node when using `drop(SplitDockPathProperty)`. As a result unmaximizing a stack of `CDockables` could destroy the stack.

### **6.3.37 SplitDockStation: update layout**

**Short** `SplitDockStation` does now always update the boundaries of its tree before dropping a `Dockable`. Wrong boundaries did lead to `Dockables` dropped at the wrong location even if the used `DockableProperty` was correct.

### 6.3.38 Path: encoding

**Short** Path now encodes its content when using `toString` and decodes contents on creation. Meaning inside a Path object the items are never encoded.

### 6.3.39 MouseFocusObserver: request focus

**Short** MouseFocusObserver no longer calls `requestFocusInWindow` if a Component does not belong to a Dockable

### 6.3.40 SplitDockStation: placeholders

**Short** SplitDockStation is now much more strict when it comes to enforcing the uniqueness of placeholders

### 6.3.41 Dockable CStation issue

**Short** If a CStation was registered at a CControl and later the same object was registered as a SingleCDockable, then CControl did not assign a unique identifier to that Station/Dockable.

### 6.3.42 SimpleDockAction: memory leak

**Short** SimpleDockAction.KeyForwarder caused a memory leak by not removing its listeners properly.

### 6.3.43 CLocation: null root

**Short** CLocations returning null as root but having an ExtendedMode are (again) supported.

### 6.3.44 SplitDockStation: locked size

**Short** When moving/dragging a Dockable from a SplitDockStation with size-locked Dockables, the locked Dockables got resized even if it was not necessary.

### 6.3.45 DockUI: stalls events

**Short** DockUI now stalls all events rather than setting the SingleParentRemover to null when updating the DockTheme of a station.

### 6.3.46 CLayoutChangeStrategy: exception

**Short** CLayoutChangeStrategy.replaceMultipleDockables could throw a NullPointerException.

### 6.3.47 CDockable: base location

**Short** CDockable.getBaseLocation did return null when a location was actually available

#### **6.3.48 Glass Extension**

**Short** Glass Extension no longer throws Exception if width or height of tab is equal to 0.

#### **6.3.49 SplitDockStation: cursor**

**Short** Cursor should no longer remain arrow when moved away from a divider of a SplitDockStation.