

BIRT API Change Control Document

BIRT API Change Control Documentation.....	1
1. Changed APIs:	1
2. Added APIs:	4
3. Removed APIs:	4
4. Miscellaneous Change Requests	6

This is a document to track the change request to the BIRT API. For each change request, the document will describe the new requirement, proposed solution and follow-up actions.

1. Introduction

To run the report engine, the user needs pass multiple settings to initialize the report engine, task and emitters. The setting methods is not consistent among different settings, it always puzzles the user.

The requested changes simplify the setting of engine, task and emitter.

In the new design, each property can be defined in two different levels: engine level or task level. If the setting is defined in the engine level, the setting will be shared among all the tasks created in the same engine. In each task, the user can override the settings in the task level setting. The task level setting is only used by the current task.

We have two kinds of settings: Application context and Emitter.

In the application context, the user can defines any properties, those properties is not handled by report engine, it will passed to the under component directly. The user can define the appContext in EngineConfig (which defines the engine level setting) or EngineTask (which defines the task level setting).

The emitter setting defines the behavior or emitters. The user can define the setting in EngineConfig through EMITTER_CONFIG (which defines the engine level setting) or in EngineTask through setRenderOption (which defines the task level setting).

The sample codes use those API is:

1. initialize the report engine.

```
EngineConfig config = new EngineConfig();
HashMap appContext = HashMap();
appContext.put("name", valueObject);
...
config.setAppContext(appContext);
HashMap emitterConfigs = new HashMap();
RenderOption defaultOption = new RenderOption();
defaultOption.setActionHandler ( action handler);
defaultOption.setImageHandler( action handler);
emitterConfigs.put("default", defaultOption);
RenderOption htmlOption = new RenderOption();
htmlOption.setXXX
emitterConfigs.put("html", htmlOption);
IReportEngine engine = new ReportEngine(config);
...
```

2. set the render option in render/runAndRenderTask.

```
IEngineTask task = engine.createTask();
HashMap appContext = new HashMap();
appContext.put("task-level-context", object);
task.setAppContext(context);
RenderOption option = new RenderOption();
Option.setOutputFormat("html");
Option.setOption("html option", option value);
Task.setRenderOption(option);
Task.render();
...
```

3. in the initializer of the emitter.

Use service.getOptions() to get the options.

4. in the ActionHandler/IImageHandler

```
String getURL(IAction action, IReportContext context)
{
    IRenderContext renderContext = context.getRenderContext();
    Object value = renderContext.getOption("option name");
    ...
}
```

2. Changed APIs:

2.1 Name = EngineConfig

Component name = report engine

Package name = Org.eclipse.birt.report.engine.api

Change Request:

1. Add getAppContext/setAppContext to get & set the engine level appContext.
2. Deprecated setEngineHome(), use setBIRTHome() instead
3. Deprecated setConfigurationVariable(), set the properties into the appContext.
4. Deprecated getConfigMap(), call getAppContext() instead.
5. Deprecated getScriptObjects(), call getAppContext() instead.
6. Deprecated addScriptableJavaObject(), add the object into appContext instead.

Proposed Solution:

2.2 Name = IRenderOption

Component name = report engine

Package name = Org.eclipse.birt.report.engine.api

Change Request:

In the render option, there are two kinds of options:

1. options used to choose the emitter. That's the IRenderOption.OUTPUT_FORMAT. The report engine use this option to choose the output emitter.
2. options used to setup the emitter. All others are used by emitter to control the output report.

Proposed Solution:

2.3 Name = RenderOption

Component name = report engine

Package name = Org.eclipse.birt.report.engine.api

Change Request:

It is the implementation of the IRenderOption. The caller can initialize this class and pass it to any emitters.

Proposed Solution:

2.4 Name = IReportContext

Component name = report engine

Package name = Org.eclipse.birt.report.engine.api

Change Request:

Add an API to get the IRenderContext.

Proposed Solution:

```
/**
 * get the render options used to render the
 * report.
 * @return
 */
IRenderContext getRenderContext( );
```

3. Added APIs:

3.1 Name = IRenderContext

Component name = report engine

Package name = Org.eclipse.birt.report.engine.api

Change Request:

This context contains all the interface used to access the render option in user defined component, such as javascript, event handler, emitter, extended item etc.

Proposed Solution:

```
/**
 * defines the context used to render the report.
 */
public interface IRenderContext
{

    /**
     * get the option defined by the key.
     * It first search the render option, then search the
     * emitter config.
     *
     * @param key
     * @return
     */
```

```

Object getOption( String key );

/**
 * get the render option setted in the task level.
 * @return
 */
IRenderOption getRenderOption( );

/**
 * get the render option setted in the engine leve.
 * @return
 */
IRenderOption getEmitterConfig( );

/**
 * the output format.
 * @return
 */
String getOutputFormat( );
}

```

4. Removed APIs:

4.1 Name = HTMLReportContext

Component name = report engine

Package name = Org.eclipse.birt.report.engine.api

Change Request:

Deprecate this class. Instead of setting the options into the context, the user should set options into the RenderOption.

Proposed Solution:

4.2 Name = RenderOptionBase

Component name = report engine

Package name = Org.eclipse.birt.report.engine.api

Change Request:

Deprecate this class, use the RenderOption directly.

Proposed Solution:

4.3 Name = HTMLEmitterConfig

Component name = report engine

Package name = Org.eclipse.birt.report.engine.api

Change Request:

Deprecated this class, the user should use RenderOption directly.

Proposed Solution:

5. Miscellaneous Change Requests

None