Transitioning to VirtualViewer Java 5.0

- Snowbound Software Corporation
- questions@snowbound.com
- (617) 607-2010

VirtualViewer Java 5.0 requires several changes to your content handler and may require updates to your custom Javascript integrations. These changes are all small-scale but necessary to keep your content handler up to date, and to keep VirtualViewer maintainable and modern through the majority of VirtualViewer releases--which do not usually modify public API.

There are five overall steps to updating your content handler and Javascript for VirtualViewer 5.0. Additional details about VirtualViewer 5.0 changes can be found in the release notes; while your organization may wish to make updates based on new features and fixes, the guide here is a starting point to get VirtualViewer 5.0 up and running. The following list provides the highlights, and there are more detailed steps and instructions in the linked sections below.

1. Update your VirtualViewer imports and dependencies to their new package locations so your content handler builds and uses the correct code (see the details)
2. Ensure that all methods intended to be called by VirtualViewer are implementations of an interface (see the details)
   Warning: this step is required for VirtualViewer to function but the content handler may still compile if it is not completed!
3. Update the logic of your `saveDocumentComponents` method so it does *not* delete a document component (a bookmarks file, for instance) when it receives a null value (see the details)
   Warning: this step is required for VirtualViewer to function but the content handler may still compile if it is not completed!
4. Throw a `VirtualViewerAPIException` instead of passing an error parameter in a `ContentHandlerResult` (see the details)
5. Modify the parameters of selected VirtualViewer Javascript API calls, which now take single objects instead of long lists of arguments (see the details)

Correct the VirtualViewer imports in your Java code

What to Do

- Use the tables below, or your IDE, to convert your content handler's `import` list to refer to the new package locations

More Details

VirtualViewer's package structure has changed substantially in version 5.0, to make public classes more clearly accessible and to organize the codebase more logically. Some import statements in your content handler code may no longer point to the right place.

For the most part, only the `import` statement will need to be updated. Some interfaces have been added as detailed in the section above, and all classes containing "FlexSnapSI" have been renamed to say "VirtualViewer" instead (for example, FlexSnapSIContentHandlerInterface is now VirtualViewerContentHandlerInterface, FlexSnapSIAPIException is now VirtualViewerAPIException, etc).

com.snowbound.contenthandler

A new package, `com.snowbound.contenthandler`, now holds all content handler-related code: interfaces, classes used in the content handler, and VirtualViewer's example FileContentHandler. Note that both `com.snowbound.snapserv.servlet.FlexSnapSISaverInterface` and `com.snowbound.snapserv.servlet.FlexSnapSIContentHandlerInterface` are now included in `com.snowbound.contenthandler.interfaces.VirtualViewerContentHandlerInterface`.

Old package location	New location in com.snowbound.contenthandler
com.snowbound.snapserv.servlet.FileContentHandler	com.snowbound.contenthandler.example.FileContentHandler
com.snowbound.snapserv.servlet.ContentHandlerInput	com.snowbound.contenthandler.ContentHandlerInput
com.snowbound.snapserv.servlet.FlexSnapSIAPIException	com.snowbound.contenthandler.VirtualViewerAPIException
com.snowbound.snapserv.transport.pagedata.VirtualViewerFormatHash	com.snowbound.contenthandler.VirtualViewerFormatHash
com.snowbound.snapserv.servlet.AllAnnotationsInterface	com.snowbound.contenthandler.interfaces.AllAnnotationsInterface
com.snowbound.snapserv.servlet.CacheValidator	com.snowbound.contenthandler.interfaces.CacheValidator
com.snowbound.snapserv.servlet.CreateDocumentInterface	com.snowbound.contenthandler.interfaces.CreateDocumentInterface
com.snowbound.snapserv.servlet.DocumentNotesInterface	com.snowbound.contenthandler.interfaces.DocumentNotesInterface
com.snowbound.snapserv.servlet.VirtualViewerContentHandlerInterface	com.snowbound.contenthandler.interfaces.VirtualViewerContentHandlerInterface
com.snowbound.snapserv.servlet.FlexSnapSISaverInterface	com.snowbound.contenthandler.interfaces.VirtualViewerContentHandlerInterface
com.snowbound.snapserv.servlet.FlexSnapSIContentHandlerInterface	com.snowbound.contenthandler.interfaces.VirtualViewerContentHandlerInterface
com.snowbound.snapserv.servlet.WatermarksInterface	com.snowbound.contenthandler.interfaces.WatermarksInterface

com.snowbound.common

Old package location	New location in com.snowbound.common
com.snowbound.snapserv.transport.P8PermissionsRecord	com.snowbound.common.permissions.P8PermissionsRecord
com.snowbound.snapserv.transport.PermissionsFactory	com.snowbound.common.permissions.PermissionsFactory
com.snowbound.snapserv.transport.PermissionsRecord	com.snowbound.common.permissions.PermissionsRecord
com.snowbound.snapserv.servlet.AnnotationLayer	com.snowbound.common.transport.AnnotationLayer
com.snowbound.snapserv.transport.pagedata.AnnotationWrapper	com.snowbound.common.transport.AnnotationWrapper
com.snowbound.clientcontentserver.ExternalReference	com.snowbound.common.transport.ExternalReference
com.snowbound.common.transport.PermissionLevels	com.snowbound.common.transport.PermissionLevel
com.snowbound.snapserv.transport.pagedata.PermissionsEntities	com.snowbound.common.transport.PermissionsEntities
com.snowbound.snapserv.transport.pagedata.FlexSnapSISnowAnn	com.snowbound.common.transport.VirtualViewerSnowAnn
com.snowbound.snapserv.transport.pagedata.RasterMaster	com.snowbound.common.util.RasterMaster

Removals

Several private VirtualViewer classes have been moved out of the `com.snowbound.common` package for clarity, so it's easier to identify what's public. The classes in `com.snowbound.common` and `com.snowbound.contenthandler` are public and available for use. Using classes in other namespaces should not be necessary and is not supported; those classes may be obfuscated.

Review your content handler's interfaces and add new required interfaces

What to Do

- Review the functions that are implemented, and not just stubbed out, in your content handler
- **For each of these functions, make sure to add the interface that describes it to the `implements` list of your content handler.** This is the most important step.
- Implement any new functions that the interface requires
- Delete unnecessary and unwanted interfaces and unnecessary stubbed-out functions

More Details

VirtualViewer 5.0 has introduced a more modular interface system to make it easier to construct content handlers without unwanted and unnecessary functionality. Any methods you do not want or need can be omitted by not implementing the required interface, instead of having to make a stubbed-out implementation.

For converting pre-5.0 content handlers to 5.0, there may now be a mismatch between the interfaces that your old content handler implements, and the actual functions that your content handler implements. This may be a silent problem that your IDE does not pick up on, so it is *imperative* to assess your content handler and add any new interfaces that are needed.

For example, the function `getAvailableDocumentIds` used to be described in FlexSnapSIContentHandlerInterface. If you have FlexSnapSIContentHandlerInterface implemented, and simply change that to VirtualViewerContentHandlerInterface without adding the new AvailableDocumentsInterface, your code may silently fail. VirtualViewer will check if your content handler implements AvailableDocumentsInterface before attempting to call `getAvailableDocumentIds`; if it doesn't implement the interface, your `getAvailableDocumentIds` function would simply never get called.

A few interfaces have been combined and removed into VirtualViewerContentHandlerInterface, but more have been separated into feature-specific interfaces.

All of the interfaces can be found in the com.snowbound.contenthandler.interfaces package and are described below:

Interface name	Description (and Changes)
AllAnnotationsInterface	Defines getAllAnnotationsForDocument (unchanged)
AnnotationsInterface	Defines annotation retrieval and modification methods (previously defined in FlexSnapSIContentHandlerInterface)
AvailableDocumentsInterface	Defines getAvailableDocumentIds (previously defined in FlexSnapSIContentHandlerInterface)
BookmarksInterface	Defines bookmark retrieval and modification methods (previously defined in FlexSnapSIContentHandlerInterface)
CacheValidator	Defines validateCache (unchanged)
CreateDocumentInterface	Defines createDocument method (unchanged)
DocumentNotesInterface	Defines document notes retrieval and modification methods (new deleteNotesContent method added)
EventSubscriberInterface	Defines eventNotification method (previously defined in FlexSnapSIContentHandlerInterface)
SendDocumentInterface	Defines sendDocument method (previously defined in FlexSnapSIContentHandlerInterface)
VirtualViewerContentHandlerInterface	Defines minimum necessary content handler methods for document manipulation (renamed from FlexSnapSIContentHandlerInterface)
WatermarksInterface	Defines watermark retrieval and modification methods (new deleteWatermarkContent method added)

Edit `saveDocumentComponents` logic for bookmarks, document notes, and watermarks

What to Do

- **Remove deletion logic from `saveDocumentComponents`. If `saveDocumentComponents` receives no content for some component, it should *not* delete that component: it should take no action.** This is the most important step.
- Implement `deleteBookmarkContent`, `deleteNotesContent`, and `deleteWatermarkContent` functions if using an interface that requires them

More Details

The content handler function `saveDocumentComponents` should no longer delete bookmarks, document notes or watermarks when it receives a null for those components. Instead, new deleteBookmarkContent, deleteNotesContent and deleteWatermarkContent methods should be implemented.

To solve an issue with our saving workflow, deleting bookmarks, document notes and watermarks now have equivalent content handler methods `deleteBookmarkContent`, `deleteNotesContent`, and `deleteWatermarkContent`, similar to the existing `deleteAnnotationContent`. If no content is provided to `saveDocumentComponents` for bookmarks, document notes or watermarks, no operation should be performed. VirtualViewer will call the equivalent delete method after `saveDocumentComponents` if necessary.

With the old logic, `saveDocumentComponents` would handle everything. If it received a `ContentHandlerInput` with a set of document notes but with bookmarks set to `null`, it assumed that bookmarks should be deleted. That is, if a component was `null` in the `ContentHandlerInput`, it should not exist on the database.

Now, that same `null` means "no action." If `saveDocumentComponents` receives a `ContentHandlerInput` where bookmarks is set to null, it should *not* delete bookmarks.

In addition to solving a saving issue, this will allow VirtualViewer to be more efficient when saving your documents - if no change is made to the above objects when saving a document, no unnecessary operations will be requested of the content handler.

ContentHandlerResult.ERROR_MESSAGE usage should be replaced with VirtualViewerAPIException

What to Do

- Locate places in your content handler where you call `ContentHandlerResult.ERROR_MESSAGE`, and replace those calls with logging statements or by throwing VirtualViewerAPIException
- Locate references to FlexSnapSIAPIException, and replace those references with VirtualViewerAPIException

More Details

ContentHandlerResult.ERROR_MESSAGE has been removed. For serious errors that should interrupt the request and inform the user, throw VirtualViewerAPIException. Errors that should not interrupt the request can still be logged with VirtualViewer by using the SLF4J Logger interface provided by SnowLoggerFactory. (VirtualViewerAPIExceptions are logged automatically.)

Replace old Javascript API calls on the client

Several Javascript API calls took more than ten parameters, which is very fragile; especially with Javascript, it's easy to forget a single parameter and produce unexpected behavior that's hard to track down. The worst offenders now take an `options` object as a parameter, which will hold named versions of the old parameters. The substance of the parameters hasn't changed, but they should be passed as part of an object instead.

For example, to manually Save As a document without its annotations but with its watermarks burned in, the old call would require eleven parameters with only four set to non-default, non-null values.

Now, the same call would be much simpler and more readable, passing in only the non-default parameters:

```
virtualViewer.saveDocument("myDocumentID.pdf",
  { "newDocumentId": "myNewDocumentID.pdf",
    "includeWatermarks": true,
    "saveAsFormat": "PDF"})
```

Changed Javascript API

What to Do

- Locate calls to `saveDocument`, `exportDocument`, `printDocument`, and `emailDocument` in your Javascript code
- Translate the passed parameters into values in a parameter object, with the keys given in the JSDoc documentation

More Details

Detailed information about the new signatures can be found in the VirtualViewer 5.0 JSDoc documentation.

Old signature	New function signature
virtualViewer.saveDocument(documentId, newDocumentId, newDisplayName, burnRedactions, includeRedactionTags, includeTextAnnotations, includeNonTextAnnotations, copyAnnotations, includeDocumentNotes, includeWatermarks, saveAsFormat, pageRangeType, pageRangeValue, copyWatermarks, options)	virtualViewer.saveDocument(documentId, options)
virtualViewer.exportDocument(exportFormat, fileExtension, includeTextAnnotations, includeNonTextAnnotations, burnRedactions, includeRedactionTags, includeDocumentNotes, includeWatermarks, pageRangeType, pageRangeValue)	virtualViewer.exportDocument(options)
virtualViewer.printDocument(documentId, printToPDF, annotations, redactions, redactionTags, watermarks, docNotes, pageRangeType, pageRangeVal)	virtualViewer.printDocument(options)
virtualViewer.emailDocument(emailFormat, includeTextAnnotations, includeNonTextAnnotations, burnRedactions, includeRedactionTags, includeDocumentNotes, includeWatermarks, pageRangeType, pageRangeValue, fromAddress, toAddresses, ccAddresses, bccAddresses, subject, emailBody)	virtualViewer.emailDocument(options)