Controls ControlsVersioning

Controls Versioning Model

Overview

There are important use cases for controls where the various related artifacts (interfaces, extensions, implementations) are developed by different people /organizations, and as those artifacts evolve over time it becomes useful to be able to specify versioning information and require certain version values. The controls programming model features a number of annotations to support this.

Defining a Version

```
package org.apache.beehive.controls.api.versioning;

public @inteface Version
{
    int major();
    int minor() default 0;
}
```

The @V'*_ersion annotation:

- Is allowed on control interfaces (annotated with @C_*'ontrolInterface).
- Defines a version of the control interface and is the basis of controls versioning.
- Is retained in interface class file (not BeanInfo!)

Specifying Version Restrictions/Requirements

```
package org.apache.beehive.controls.api.versioning;

public @interface VersionRequired
{
    int major;
    int minor default -1;
}
```

The @V'*_ersionRequired annotation:

- Is allowed on control extensions and controls field declarations.
- Defines the minimum versions of the control interface that this extension or field requires.
- Major version mandatory, minor is optional (default is to not require any specific minor ver).
- When extension or client is compiled, version requirement is enforced against the control interface found at that time.
- When extension's control bean is is classloaded, version requirement is statically enforced against the control interface found at that time.

```
public @interface VersionSupported
{
    int major;
    int minor default -1;
}
```

The @V_*'ersionSupported annotation:

- Is allowed on control implementations.
- Defines the maximum versions of the control interface that this implementation supports.
- Major version mandatory, minor is optional (default is to support all minor versions).
- When implementation is compiled, version requirement is enforced against the control interface found at that time.
- When implementation is classloaded, version requirement is statically enforced against the control interface found at that time.

Open Issues

- Static runtime enforcement may run into problems in complex classloader situations (where interfaces and impls come from different classloaders). Perhaps add a way to disable runtime checks?