IBM Cúram Social Program Management

Cúram Development Compliancy Guide

Version 6.0.5

IBM Cúram Social Program Management

Cúram Development Compliancy Guide

Version 6.0.5

Note

Before using this information and the product it supports, read the information in “Notices” on page 29

Revised: May 2013

This edition applies to IBM Cúram Social Program Management v6.0 5 and to all subsequent releases unless otherwise indicated in new editions.

Contents

Figures

Tables

Chapter 1. Introduction

1.1 Purpose

This document provides guidelines on how to build custom functionality in a compliant manner.

Note that from version 6.0.3, some of these guidelines have changed. Whereas all application customization mechanisms continue to be supported for customers who have already used them, some of them are now discouraged for new development.

1.2 Intended Audience

This document is intended to be read by designers and developers of project teams who are building Cúram applications.

1.3 Prerequisites

A working knowledge of the application development environment is needed to read this document. References to pertinent development documentation are provided throughout this guide.

Chapter 2. Developing Compliantly with Cúram

2.1 Overview

This chapter describes important considerations that need to be taken into account in order to develop compliantly with Cúram. These considerations are essential for Support to be able to assist customers in applying their own customizations to the application. By following these considerations, customers will also find it easier to upgrade to future versions.

2.2 Starting a New Project

When starting a new project, it is important to understand the development directory structure. It is also important to put it under source code control.

2.2.1 Understand the Development Directory Structure

Knowledge of the development directory structure is required to understand where development artifacts are located, how they are organized, and where to store changes to these artifacts. Note that to access the development directory structure, you must first install a development version of the application.

The following list describes the directories into which the client and server development artifacts are installed:

- Client development artifacts are installed into the `webclient` directory. For details on how to develop client applications compliantly, see the *Cúram Web Client Reference Manual*
- Server development artifacts are installed into the `EJBServer` directory.

Within both the `webclient` directory and the `EJBServer` directory, there is a `components` subdirectory, which has a further subdirectory called `custom`. The `custom` subdirectory is where all project specific development artifacts should be placed. The other `components` subdirectories contain all of the application development artifacts delivered with the product.

Important: The `custom` folder contains a starter structure for first usage and is referred to throughout developer documentation as the area in which all artifacts should be developed. It should be noted that this is not enforced and it is a project choice to develop within this component or create a new named component appropriate for your project.

Within the `EJBServer\components\custom\model` directory, there is a starter model file and some model fragments.

2.2.2 Source Code Control

To keep track of all changes to source artifacts, the development directory structure should be put under source code control. Once under source code control, all development artifacts should be tagged. Ensure that the tag refers to the version of the application. At any point, it will then be possible to produce a report, using diff functionality, of all files that have been added or changed to implement project functionality. This report is useful when taking on a new release of the application.

Note that from version 6.0.3, changes have been made to how Java™ source code is delivered. See section 4 below for more information.

2.3 Changing Server Source Artifacts

There are many types of server artifact, some of which are application classes. Some of these are represented in an application model. Other Java interfaces are "handcrafted". There are application implementations of both these categories of class, and it is important to be able to distinguish between the two. Whereas it is possible to change aspects of a modeled interface by changing the model and regenerating code, it is not possible to change a handcrafted interface.

Modeled interfaces:

- Appear in the application UML model

Handcrafted interfaces:

- Do not appear on the application UML model
- Appear in the component directories of your development environment
- Cannot be customized
- Contain the `@ImplementedBy` Google Guice annotation to indicate of the application implementation class

Some components may contain interfaces which do not fall into either of the above categories, but these will always be described in component-specific documentation. Both modeled and handcrafted application interfaces may have implementations which can be customized.

For details on how to implement source artifacts, see the *Cúram Server Developer's Guide* for implementations of modeled interfaces, and the *Persistence Cookbook* for implementations of handcrafted interfaces; it is necessary to look at the implemented interface to determine the category. The recommendations on how to change Server Source Artifacts have changed with version 6.0.3. Note that the recommendations contained in this document (the Cúram Development Compliancy Guide) are definitive.

2.3.1 Write Source Code for New Methods and Classes

New customer-specific classes, classes which wrap existing classes, or in a limited set of circumstances new subclasses of existing classes should be written in new source files. All new source files should be placed within the `source` subdirectory of the `EJBServer\components\custom` directory.

For modeled classes, the generated class hierarchy will dictate the package structure of the new source files. See the *Cúram Server Developer's Guide* for information on modeling new classes and replacing existing implementations.

For handcrafted implementations, it is up to you how the new class is packaged. See the *Persistence Cookbook* for information on configuring new subclasses using Google Guice.

2.3.2 Changing Cúram Express Rules (CER) Rule Sets

The CER Editor stores its rule sets on the database rather than the file system. Any rule sets shipped in the core component must NOT be customized. Solutions may have their own compliancy statements about their rule sets.

2.3.3 Extending Codetables

Note that documentation has now been provided to indicate which codetables are safe to extend, and which require customers to ask Support before customizing. A list of codetables which cannot be extended without contacting Support is provided in the project documentation directory structure for every installation (in a folder called RestrictedCodeTables). If you want to customize a codetable which is listed in this list, you should raise a Support case.

2.4 Source Code and APIs

All application Java functionality is now being distributed as pre-built jar files. This had always been the case for Enterprise Modules introduced since version 5.0 (for which source code was never shipped) but has now become universally true. Application functionality will now only be regenerated and rebuilt in a customer installation if required by the use of customer extension mechanisms. This means that the customer build process will no longer need to rebuild the entire Java source code base; only project-specific source code and any dependent regenerated Java source code will now need to be rebuilt.

For a limited number of key functional areas from version 6.0.3 onwards, Java source code is no longer distributed in any form. Source code for the remainder of the application continues to be shipped (as 'sample'), but for documentation purposes only - this code is not directly involved in the build process from version 6.0.3. This sample source code is distributed in jar files on a per-component basis as follows: `EJBServer\components\<component name>\sample\src.zip`. The built versions of each components can be found in the following location: `EJBServer\components\<component name>\lib\<component name>.jar`.

Also from version 6.0.3, class operations have been marked as Internal or External via annotations.

External operations form the official API to the application going forward, which customers are encouraged to use and invoke from their own code.

Important: Classes with no annotations are Internal by default

2.4.1 Internal APIs

Whereas it is possible to invoke and subclass Internal APIs from custom code, this is discouraged from version 6.0.3. Such APIs are annotated with `@Accesslevel(INTERNAL)`.

Important: 'Discouraged' in this context means that their use continues to be supported, but that such APIs may be changed or removed in future releases, once a minimum notice period of 1 year has been given to customers in respect of any such change or removal.

Note: No such notice is being given for any of the APIs marked as Internal in version 6.0.3. (i.e. there are no current plans to change any of the APIs marked as Internal in 6.0.3), and so there should be adequate time for customers to plan any such migrations.

Existing customer references to APIs which are marked as Internal from version 6.0.3 will continue to function as before, with the exception that discouraged warnings will be generated within Eclipse projects which have such dependencies.

Projects should endeavor to move away from such dependencies on Internal APIs over time, and should not introduce new dependencies on them (within reason - depending on where a customer project is in its design/development process, it may be inevitable in the short term). Most existing customers will see discouraged references reported after taking on version 6.0.3 or later versions, and it is not expected that customers fix these immediately as part of the take on activity. As mentioned above, this will not affect their support entitlements.

Note that as with previous versions of the application, some Internal APIs have been configured to produce 'access restriction' errors in Eclipse if referenced (these APIs are annotated with `@Accesslevel(RESTRICTED)`), and such references will not be supported in customer projects. These APIs have always been Internal, and were never supported for customer use; it will be obvious which are which - access restricted APIs produce Eclipse errors, discouraged APIs produce Eclipse warnings.

2.4.2 External APIs

External APIs can be referenced directly by customer projects. Such APIs are annotated with `@Accesslevel(EXTERNAL)`. Javadoc is provided for all External APIs on a per component basis, and this can be found at EJBServer\components\<component name>\doc\api.zip. Note that some components may not have any Javadoc as they have no External APIs. Only classes that are documented in JavaDoc should be referenced from customer code; referencing other classes will produce discouraged warnings or access restricted errors and are not supported.

Note also that, as with all APIs, it is expected that those marked as External will evolve over time (while remaining backward compatible). If you have a requirement which you feel cannot be fulfilled through a combination of the use of External APIs and allowed extension mechanisms, you should raise these through Support. If appropriate, a new API, customization hook, strategy pattern or configuration-based approach will be made available, and such new APIs can be delivered in Feature Packs. Alternatively, an existing Internal API may in some circumstances be redesignated as External if appropriate.

2.4.3 Extension Mechanisms

The removal of source code from the areas of key functionality referred to above has resulted in a change to the recommended approach to using extension mechanisms on customer projects. Previously, if customers wished to use the various application extension mechanisms (e.g. extension classes, subclass with and without replace, aggregation), they could search across the codebase to see where and how target classes were being invoked within application code. They could then make an assessment of the functional effects of the extension being considered.

From 6.0.3, customers will no longer have the source code for some areas of key functionality, and in addition a large number of APIs have been marked as Internal. The following section summarizes the change in recommended extensions practices for customer projects.

Note that this section only refers to restrictions on extending application artifacts. All extension mechanisms can continue to be used on customer-defined classes, and all such artifacts can of course be External in nature, and invoked from any other part of a customer implementation.

Important: This section just provides a high-level summary. Full details of which mechanisms are allowed on which class types from version 6.0.3 are provided in Appendix B, “Discouraged Extension Mechanisms,” on page 15. Where mechanisms have been discouraged, this appendix will where appropriate recommend alternative mechanisms to be employed by customers.

2.4.3.1 Entity Classes

With some exceptions, direct customer use and modification of application Entity classes is now discouraged. In many cases, application Entity class operations have direct Facade-layer equivalents which have been marked as External, and these can be used by customers. Addition of stereotyped and non-stereotyped operations to application Entities is however still allowed, as is the setting of a number of Entity options.

Prior to version 6.0.3, attributes could be added to application Entity classes using extension. However, with source code being removed for areas of key functionality, customers will no longer have visibility as to whether attributes added via extension classes will be mapped to external APIs. For this reason, adding attributes to application Entity classes is now discouraged.

Customers wishing to add data to application screens should add new customer-specific Entity classes, and should wrap External application maintenance operations in their own process classes to maintain both tables atomically. Application screens can then be changed to point to the new process classes.

Note: Entities representing Evidence Types are an exception to this rule. Customers can continue to add attributes to such application Evidence Entities using extension, as this is required by the Evidence Generator.

In version 6.0.3, application Evidence Entities have incorrectly been marked as Internal; this will be corrected in a subsequent release. For now, customers using extension on Evidence Entities to add attributes may see discouraged warnings in Eclipse relating to these classes; these specific warnings can be ignored.

This note only applies to Entities which represent Evidence Types and not for any other application Entity class.

2.4.3.2 Domain Definitions

In general, customer use and overriding of application Domain Definitions is still allowed. However, changing the fundamental type of a Domain Definition is now discouraged, as is changing of a number of codetable-related options.

2.4.3.3 Struct Classes

Application struct classes are all essentially External in nature, in that they can be referenced in customer-specific functionality.

Customers are discouraged from directly creating aggregations from application structs to any other struct (as they no longer have full visibility on where these application structs are being used). Customers can however continue to use aggregation to include application structs in their own project-specific structs.

2.4.3.4 Other Modeled Classes

For other modeled classes in the application (such as Process, Facade, WSInbound and WebService), the use of all extensions mechanisms is now discouraged.

Prior to version 6.0.3, Subclass with Replace was a commonly used mechanism for adding and changing operations on application Process and Facade classes. As with extension of application Entity classes, however, this is now potentially unsafe, in that customers will no longer necessarily have full visibility as to where such classes are used.

Similar to with Entity classes, customers should instead model and code their own Process, Facade or WSInbound classes, either wrapping existing External APIs, or implementing new functionality. For Facade operations, affected UIM pages can be repointed at the new Facade operations if desired.

2.4.3.5 Non-Modeled Classes

Some components contain non-modeled classes. For these classes, the usage of each External interface or class is described in the Javadoc for the class.

Some non-modeled classes come with Eclipse access restrictions in place to provide customers with guidance in relation to which APIs they can and cannot call or customize. Certain classes and packages are marked as restricted; these classes must not be used as they are internal classes that can change over time. Access restrictions should not be removed from the Eclipse.classpath file as this may result in the consumption of restricted classes which can cause problems during upgrades.

Some non-modeled components contain package protected classes; these classes should not be used in custom code. Customers must not place any custom code in the same package structure in order to call or reference package protected classes.

Many non-modeled APIs are not directly customizable. Only interfaces/classes tagged with the `@Implementable` annotation can be extended or implemented. Such classes will have JavaDoc detailing how to customize or implement them. Non-modeled classes that are not tagged with the `@Implementable` annotation must not be extended or implemented as new operations may be added overtime which may cause upgrade impact.

For classes tagged with the `@Implementable` annotation, the typical customization mechanisms for these types of class are events and strategies.

Events allow customers to add custom logic at various points in the application. For details on how to add event listeners, please refer to the Persistence Cookbook. Event classes are typically named ‘xxxEvent', so they can be easily identified.

Strategy patterns allow customers to change the default behavior of certain functions within the application. Each strategy class has a default implementation provided; however customers can choose to override the default implementation of any of the strategy operations through the use of Guice bindings. The further details on using Guice bindings, please refer to the Persistence Cookbook. Strategy classes are typically named ‘xxxStrategy', so they can be easily identified.

Note: For further compliance details on a per-component basis, please refer to Appendix A, “Component Compliance Details,” on page 13.

2.4.4 Summary

In summary:

Where you want to reference an application class in your custom code:
- If the class is External, you are allowed to reference it.
- If the class is Internal, you are supported in referencing it in your existing code but discouraged from doing so. You should not reference it in new code.
- If the class is Access Restricted, you are not supported in referencing it.

Where you want to customize an application class:
- If the class is Modeled, refer to Appendix B for details of allowed customizations
- If the class is Non-Modeled, refer to its JavaDoc and/or any configuration/development guide for its parent component for details of customization points.

2.5 Avoiding Common Compliancy Pitfalls

This section describes compliancy issues that can arise and provides rules-of-thumb for avoiding these issues. Following the rules-of-thumb presented in these sections from the early stages of a project is relatively easy. However, if they are not followed, they can result in serious disruption later on and fixing them can be both costly and difficult.

2.5.1 Use Project-specific Prefixes in Artifact Names

You should prefix all new source artifact names (model classes, source files, messages, message files, etc.) with a relevant acronym or abbreviated word. Use the same acronym or abbreviated word throughout. As the project progresses, this will make project additions to core artifacts more obvious. This distinction becomes very useful as the development effort grows. Generally most projects will be described by some kind of acronym. This acronym is a good candidate to use as the prefix.

Using a project-specific prefix prevents naming collisions from occurring between new artifacts that you add and new artifacts that Cúram adds over time; naming collisions can be costly and difficult to fix when they occur.

For example, consider taking on a Service Pack and discovering that one of your custom database field additions has the same name as a new application field that does not have the same business meanings or data-type. Alternatively, consider taking on a Service Pack and discovering that a new application Codetable Item has been added that conflicts with a custom Item that you also added with the same name, but a different meaning. These types of collisions can be avoided by ensuring that you always name new, custom artifacts with a consistent prefix.

Some artifact types have more than one identifier and these must be taken into account when naming them. Entity classes and Codetable Items are examples of this. A custom Entity class has a Table Name that shares the same flat namespace (the database schema) as application Tables and must have a unique Table Name within that namespace. It also has a Java class name, which shares a hierarchical namespace (package structure) with application Java classes. Likewise, a custom Codetable Item has both a value and a Java identifier – and both share a flat namespace with application items in the same Codetable.

It is important to note that the use of project specific prefixes does not apply where you are overriding an application artifact, as the override mechanism is usually based on naming your custom artifact with exactly the same name as the application artifact that it overrides.

Additional considerations:

- Identifiers come in many flavors – e.g. a filename, an XML ID, a Java class name or a combination thereof.
- A short prefix is advisable as there may be places where name lengths are restricted (e.g., certain types of database identifiers).

Note: In addition to source artefacts, it is also important to consider identifier values which may conflict with values used by IBM.

The API `TransactionInfo.setFacadeScopeObject` and `TransactionInfo.getFacadeScopeObject` enables developers to access objects which are associated with the current transaction. When using this API, to ensure that any of your data for the transaction does not conflict with data belonging to IBM you should use a `String` as your object identifier and prefix this string with an appropriate word as described above.

2.5.2 Use Numeric Identifiers in Custom Initial, Demo Data

Pre-defined initial and demo data is loaded into an application database via DMX files. This data is installed into the database when a system is first set-up, or when a system is upgraded. A set of initial and demo data is provided in the application. Customers may also need to add their own initial and/or demo data.

In order to avoid clashes with the initial and demo data that is shipped in the application and with data created by the runtime system, it is important that the identifiers (e.g., primary keys) for customer initial and demo data are drawn from reserved ranges. Therefore, a set of ranges has been reserved for customer use.

2.5.2.1 Reserved Ranges

Projects should use identifiers (primary keys) in their custom initial and demo data that are drawn from the following reserved ranges:

- Non-human readable primary keys: 45,000 to 49,999 (inclusive)
- Human readable primary keys: 11,521 to 12,799 (inclusive)
- Rule sets: 4,500 to 4,999 (inclusive)

Customers that have already used identifiers from outside these ranges will be assisted in addressing this in advance of performing their next upgrade.

2.5.2.2 Large Data Sets

From time to time it may be necessary to generate very large sets of data. For example, this may be required for load testing. In these cases, the number of records required would far exceed the allocated key ranges documented here. In this situation, a different approach should be taken.

Instead of using keys from the allocated ranges, the key server should be used to generate the key values required. If this data will be imported into a re-built database, the final value of the key set should also be extracted and loaded into the key set table, replacing the initial key set value supplied in the application. If you have any questions around this process, please contact Support for further information.

2.5.2.3 Codetables Exception

Please note that the above statement does not apply to code tables.

2.5.3 Never Make In-Place Modifications to Application Files

Service Packs and Emergency Patches need to be able to safely move, restructure or overwrite application files. If you modify these files, Service Packs or Emergency Patches can overwrite them without notice. There is no guarantee that these changes will be compatible with the modifications you made, so re-applying the in-place changes afterwards may not be possible.

There are a very small number of exceptions to this rule and these are listed below:

- EJBServer
  - `/project/config/datamanager_config.xml`
  - `/project/config/deployment_packaging.xml`
  - `/project/properties/Bootstrap.properties`
  - `.classpath`
  - `.project`
- Webclient
  - `/JavaSource/curam/omega3/ApplicationConfiguration.properties`
  - `/JavaSource/curam/omega3/i18n/CDEJResources.properties`
  - `.classpath`
  - `.project`

2.5.4 Never Create Dependencies on Sample or Demo Artifacts

Sample and Demo artifacts are off-limits for custom dependencies, i.e., references-to APIs, UIM files, codetables, message files, etc. in such components from custom code. These artifacts are subject to change without notice.

Different product areas in Cúram have taken different approaches to marking off artifacts as Internal/Sample/Demo, so this guide cannot give a concise statement of how to identify them. However, there are a few reliable rules of thumb:

- Artifacts whose name, code package, model package or file path contain the words 'Internal', 'Sample', or 'Demo' (or obvious derivatives of those words)

If in doubt, contact Support.

Important: The `CPMSample` folder is internal; all code and artifacts within this folder can change without any notice. If customers wish to use functionality within CPMSample, they will need to duplicate it in their code base.

2.5.5 Reflecting Changes to Dynamic Artifact Types Back to Development System

If you modify dynamic artifact types on production or test systems, you should always ensure that these modifications are reflected back to the development system.

Various 'Dynamic' development artifacts exist in the application that can be modified at runtime on a production or test system (e.g., codetables, workflows, etc.). Runtime changes to these artifacts should always be synchronized back to the development codebase so that concurrent development changes can be integrated with these runtime changes prior to deployment.

Concurrent changes to these artifacts may happen during routine project milestone development, or when taking on Service Packs or doing Major/Minor version upgrades. In every case, there must be one central place where concurrent changes are merged together and validated and this is the development codebase. The System of Record for these artifacts is the development codebase.

2.5.6 Don't Create New Dependencies on Internal APIs

From version 6.0.3, customers should avoid invoking or customizing application classes and operations marked as 'Internal', as such APIs may change in subsequent versions of the application.

Appendix A. Component Compliance Details

A.1 Introduction

This section contains individual per-component compliance information.

Important: Unless otherwise indicated, for all components (whether listed here or not) it can be assumed that the general following general compliance statements apply:

Where you want to reference an application class in your custom code:
- If the class is External, you are allowed to reference it.
- If the class is Internal, you are supported in referencing it in your existing code but discouraged from doing so. You should not reference it in new code.
- If the class is Access Restricted, you are not supported in referencing it.

Where you want to customize an application class in your custom code:
- If the class is Modeled, refer to Appendix B for what you are allowed to do
- If the class is Non-Modeled, refer to its JavaDoc (EJBServer\components\<component name>\doc\api.zip) for what you are allowed to do.

Table 1. Component Compliance Details.

This table lists the components with Non-Modeled APIs.

Component	Details
Cúram Client Development Environment	For further guidelines on how to customize/use this component please refer to the *Cúram Web Client Reference Manual.* Please note that files from the `CuramCDEJ` folder will be copied to temporary build folders during the application build process. The presence of such files outside of the `CuramCDEJ` folder does not make them available for customization.
Cúram Server Development Environment	This component's Javadoc details all customization points and External APIs. Only classes that are documented in JavaDoc should be referenced from customer code; referencing other classes will produce discouraged warnings or access restricted errors and are not supported(Cúram's cryptographic functionality is not supported for customer use beyond the documented usage in the *Cúram Server Developer's Guide* and *Cúram Security Handbook*.). The bin directory of this component contains Apache Ant build scripts that must not be modified directly. Updates to these scripts can be made by creating new custom ant scripts and using the Ant inheritance functionality. The drivers folder of this component contains database drivers used to access the application database. If necessary, these drivers may be replaced with the relevant driver for the database being used, provided the database is a supported database version as specified in the *Cúram Supported Prerequisites.* **Note:** If a problem arises with a driver that has not been shipped in the product (i.e. that has not been tested and verified for use with the application), the customer may be requested to replace the driver with a version that has been tested, while the specific issue is raised with the third party vendor. Please note that files from the `CuramSDEJ` folder will be copied to temporary build folders during the application build process. The presence of such files outside of the `CuramSDEJ` folder does not make them available for customization.
Cúram Administration Suite	Note that from version 6.0.3, the compliance statement for classes in the Cúram Administration Suite is no different from those in any other component. External APIs in the Administration Suite can be wrapped and invoked from custom code.
Persistence Infrastructure	The Persistence Infrastructure cannot be customized. Customers must not place any custom code in Persistence Infrastructure's code packages (curam.util.persistence and all sub-packages). For further information on how to use these APIs please read the *Persistence Cookbook.*
CER Infrastructure	The compliancy statement for CER Infrastructure can be found in the *Cúram Express Rules Reference Manual.* CER entities (i.e., any entity whose name is prefixed by the word Creole) should be considered Internal and subject to change, and customers should not update them or query them except via the CER API or DMX files.

Table 1. Component Compliance Details (continued).

This table lists the components with Non-Modeled APIs.

Component	**Details**
Dependency Manager	The Dependency Manager encompasses all server artefacts in the `curam.dependency` code package and all its sub-packages. The following components contribute to the Dependency Manager code package: • the CER Infrastructure; and • the core application. The Dependency Manager cannot be customized in any way. All Dependency Manager APIs are for internal use only. The compliancy statement for the Dependency Manager can be found in the *Cúram Express Rules Reference Manual*.
Eligibility and Entitlement Engine API	For guidelines on how to configure and customize this component, please read the *Inside Cúram Eligibility and Entitlement Using Cúram Express Rules Guide*.
Evidence Generator	The Evidence Generator is application infrastructure that is shipped as part of the Tools directory structure (EGTools). For more information on using the Evidence Generator, see the *Cúram Evidence Generator Specification*.
DocMaker	No part of the DocMaker tool may be customized.
Pod Infrastructure	Pod Infrastructure is shipped in the widget-inf.jar and widget-utility.jar files. The Pod Infrastructure cannot be customized. Pod-Loaders cannot be customized. For further information on developing Pods see the *Cúram Pod Developer's Guide*.
Funded Program Management	For guidelines on how to customize this component, please read the *Funded Program Management Developer Guide* and the component's javadoc..
Cúram Incidents	For guidelines on how to customize any Incident Entities or replacing any Incident implementation please read the *Persistence Cookbook* and the component's javadoc.
Cúram Citizen Context Viewer	For further guidelines on how to customize this component please refer to the *Cúram Citizen Context Viewer Configuration Guide* and the component's javadoc.
Cúram Advisor	The following server components are delivered with Cúram Advisor: Advisor.
Cúram Common Intake	The following server components are delivered with Cúram Common Intake: Intake, PCR, CREOLEProgramRRecommendation, ReferralsLite and CPMReferralsLite
Inbox	For guidelines on how to configure and customize this component, please read Part VI of the *Cúram Workflow Reference Guide*.
Cúram Waitlists	For guidelines on how to customize this component, please read the *Cúram Waitlist Customization Guide* and the component's javadoc.
IBM Cúram Business Intelligence and Analytics	For guidelines on how to customize this component, please read the *Cúram Business Intelligence Reporting Developer Guide*
IBM Cúram Social Enterprise Collaboration	The following server components are delivered with Social Enterprise Collaboration: SocialEnterpriseCollaboration, CaseParticipantIndex and ClientAccess.
IBM Cúram Universal Access	For further guidelines on how to customize this component please refer to the *Cúram Universal Access Developers Guide* and the component's javadoc.
IBM Cúram Outcome Management	The following server components are delivered with Cúram Outcome Management: AssessmentPlanning, AssessmentPlanningCPM, DecisionAssistAssessments and SimpleOutcomeManagement
IBM Cúram Provider Management	For guidelines on how to customize this component, please read the *Cúram Provider Management Developer Guide* and the component's javadoc.
IBM Cúram Youth Services(CYS)	For guidelines on how to customize any CYS Entities or replacing any CYS implementation please read the *Persistence Cookbook* and the component's javadoc.
IBM Cúram Child Care (CCC)	For guidelines on how to customize any CCC Entities or replacing any CCC implementation please read the *Persistence Cookbook* and the component's javadoc.

Appendix B. Discouraged Extension Mechanisms

B.1 Introduction

As described earlier, many of the mechanisms previously recommended in versions prior to 6.0.3 as a means of extending or replacing application classes are now discouraged. This appendix details which mechanisms are allowed and discouraged when applied to which class types. It also suggests what to do if you find that a mechanism/class type combination that you want to employ is now discouraged.

B.2 Extension Classes

B.2.1 Entity

Table 2. Extension Classes as Applied to Entity Classes

Action	Model Option	Discouraged ?	Alternative
Add a stereotyped entity Operation (e.g. <<ns>>, <<nsreadmulti>>)	None	Discouraged	Rather than using an <<extension>> class, add the stereotyped operation through the use of subclass without replace.
Change an entity Operation (e.g. parameters)	None	Discouraged	Create a new stereotyped operation with the desired structure using subclass without replace. If you feel you have a valid need to change the structure of an application Entity operation, please raise a Support case.
Change an Entity operation option	Auto ID Field Auto ID Key No Generated SQL Optimistic Locking Order By SQL Where	Discouraged	Create a new stereotyped operation using subclass without replace. If you feel you have a valid need to change these options on application Entity operations, please raise a Support case.
	Database Table-level Auditing	Discouraged	This option is settable via runtime properties. See section 12.3.6 of the Cúram Server Modeling Guide for more information on how to do this.
	On Fail Operation Post Data Access Operation Pre Data Access Operation Treat Readmulti Max as Informational Exception Readmulti Max Records Returned	Discouraged	Customers should only change these options on application Entity operations by using Subclass with Replace.
Change an Entity class option	Enable Validation	Discouraged	Customers should only change this option on application Entity operations by using Subclass with Replace.

Table 2. Extension Classes as Applied to Entity Classes (continued)

Action	Model Option	Discouraged ?	Alternative
	Abstract Allow Optimistic Locking No Generated SQL Replace Superclass	Discouraged	If you feel you have a valid need to change these options on application Entity operations, please raise a Support case.
	Audit Fields Last Updated Field	Allowed	Currently only supported via Extension classes, and this will continue to be the case from 6.0.3
Add an Entity attribute	None	Discouraged	Customers wishing to add data to application screens should add new customer-specific Entity classes, and should wrap Cúram CRUD operations in their own process classes to maintain both tables atomically. Cúram screens can then be changed to point to the new process classes
Change an Entity attribute option	Allow Nulls	Discouraged	If you feel you have a valid need to change this option on application Entity attributes, please raise a Support case.

B.2.2 Struct

Table 3. Extension Classes as Applied to Struct Classes

Action	Model Option	Discouraged ?	Alternative
Add an attribute to a struct	None	Discouraged	Create a new project-specific struct, and aggregate the application struct from the project-specific struct to the application struct (not the other way around). Use the new 'composite' struct in required customer-specific functionality.
Change a struct attribute	None	Discouraged	Create a new project-specific struct, and aggregate the application struct from the project-specific struct to the application struct (not the other way around). Use the new 'composite' struct in required customer-specific functionality. If you feel you have a valid need to change an attribute of an application struct, please raise a Support case.
Change a struct option	Audit Fields	Discouraged	If you need to propagate Audit Fields from an Entity through to a screen, you will need to create new stereotyped operations which maintain the Audit Fields, create a new Facade which wraps the existing Entity CRUD operations and calls the new stereotyped operations, and update any UIM pages as required.

B.2.3 Process, Facade, WebService, WSInbound

Table 4. Extension Classes as Applied to Other Modeled Classes

Action	Model Option	Discouraged ?	Alternative
Change a class option	Abstract Generate FIDs Replace Superclass WS Binding Style WS Is XML Document Document Type Generate Facade Bean Provider Name Request Handlers Response Handlers Validate Request XML Document XML Schema	Discouraged	If you feel you have a valid need to change these options on application Process, Facade, WebService or WSInbound classes, please raise a Support case
Add an operation	None	Discouraged	This was never encouraged via extension classes, in that it would have required customers to perform in-place modification of application Java code. If you want to add an operation to a Process, Facade, WebService or WSInbound class, wrap the class and operation in your own project-specific class and operation.
Change an operation (e.g. operation visibility)	None	Discouraged	Create a new operation in a project-specific class, wrapping External APIs of the application class functionality if appropriate. If no appropriate extension point exists, but you feel you have a valid need to change the functioning or structure of an application operation, please raise a Support case.
Change an operation option	Audit BI Calls	Discouraged	This option is settable via runtime properties. See section 12.3.6 of the Server Modeling Guide for more information on how to do this.
	Business Date Field Bytes Message Encoding Character Set Generate Security Is XA Transactional Message Type Queue Connector Factory JNDI Name Reply Queue JNDI Name Response Message Timeout Shadow Type Transactional Transmission Queue JNDI Name	Discouraged	If you feel you have a valid need to change any of these options on application Process, Facade, WSInbound or WebService operations, please raise a Support case.

Table 4. Extension Classes as Applied to Other Modeled Classes (continued)

Action	Model Option	Discouraged ?	Alternative
	Secure Fields	Discouraged	Customers wishing to alter which fields of an application operation are to be treated as Secure should wrap the operation in their own Facade class and operation, and set the Secure Fields option on this new operation to the desired setting. Affected UIM screen definitions should be repointed at the new operation if required.
Change an operation parameter option	Mandatory Fields	Discouraged	Customers wishing to alter which fields of an application operation are to be treated as Mandatory should wrap the operation in their own Facade class and operation, and set the Mandatory Fields option on this new operation to the desired setting. Affected UIM screen definitions should be repointed at the new operation if required.

B.3 Subclass With Replace

B.3.1 Entity

Table 5. Subclass With Replace as Applied to Entity Classes

Action	Model Option	Discouraged ?	Alternative
Add a stereotyped Entity operation (e.g. <<ns>>, <<nsreadmulti>>, etc.)	None	Discouraged	Rather than using Subclass with Replace, add the stereotyped operation through the use of Subclass without Replace. This will ensure that your subclass (and thus your new stereotyped operations) will be treated as 'External', and that you won't get discouraged warnings in Eclipse when you reference them. Note that you will continue to get discouraged warnings if you directly reference stereotyped operations in the base Entity, as these are Internal - this is by design.
Add or Change a non-stereotyped Entity operation	None	Discouraged	Rather than using Subclass with Replace, add a non-stereotyped operation through the use of Subclass without Replace. This will ensure that your subclass (and thus your new non-stereotyped operations) will be treated as 'External', and that you won't get discouraged warnings in Eclipse when you reference them. Note that you will continue to get discouraged warnings if you directly reference operations in the base Entity, as these are Internal - this is by design. Customers are discouraged from providing new implementations for non-stereotyped application Entity operations.
Change the structure of an Entity operation	None	Discouraged	Create a new stereotyped operation using subclass without replace. If you feel you have a valid need to change the structure of an application Entity operation, please raise a Support case.
Change an Entity operation option	Auto ID Field Auto ID Key No Generated SQL Optimistic Locking Order By SQL Where	Discouraged	Create a new stereotyped operation using subclass without replace. If you feel you have a valid need to change these options on application Entity operations, please raise a Support case.
	Database Table Level Auditing	Discouraged	This option is settable via runtime properties. See section 12.3.6 of the Server Modeling Guide for more information on how to do this.

Table 5. Subclass With Replace as Applied to Entity Classes (continued)

Action	Model Option	Discouraged ?	Alternative
	On Fail Operation Post Data Access Operation Pre Data Access Operation	Allowed (Partially)	Customer are still allowed to implement application Entity exit points. If customers want to perform processing in exit points for which there is a default implementation, the default implementation must be invoked at the beginning of the customer exit point implementation (i.e. there must be a call to 'super()' at the beginning). Customers are not allowed to switch off application exit point implementations.
	Treat Readmulti Max as Informational Exception Readmulti Max Records Returned	Allowed	
Change an Entity class option	Enable Validation	Allowed (Partially)	Customer are still allowed to implement application Entity exit points. If customers want to perform processing in exit points for which there is a default implementation, the default implementation must be invoked at the beginning of the customer exit point implementation (i.e. there must be a call to 'super()' at the beginning).
	Abstract Allow Optimistic Locking No Generated SQL	Discouraged	If you feel you have a valid need to change these options on application Entity operations, please raise a Support case.
	Audit Fields Last Updated Field	Discouraged	Use Extension classes to override these options on an application Entity class.
	Replace Superclass	Allowed (Partially)	Implicitly allowed to support other 'Allowed' actions described in this section

B.3.2 Process, Facade, WebService, WSInbound

Table 6. Subclass With Replace as Applied to Other Modeled Classes

Action	Model Option	Discouraged ?	Alternative
Change a class option	Abstract Generate FIDs Replace Superclass WS Binding Style WS Is XML Document Document Type Generate Facade Bean Provider Name Request Handlers Response Handlers Validate Request XML Document XML Schema	Discouraged	Create a new operation in a project-specific class, wrapping External APIs of the application class functionality if appropriate, and setting the appropriate options on the new class. If you feel you have a valid need to directly change these options on application Process, Facade, WebService or WSInbound classes, please raise a Support case.
Add an operation	None	Discouraged	Create a new operation in a project-specific class, wrapping External APIs of the application class functionality if appropriate.
Change an operation	None	Discouraged	Create a new operation in a project-specific class, wrapping External APIs of the application class functionality if appropriate. If you feel you have a valid need to directly change the structure of operations on application Process, Facade, WebService or WSInbound classes, please raise a Support case.
Change an operation option	Audit BI Calls	Discouraged	This option is settable via runtime properties. See section 12.3.6 of the Server Modeling Guide for more information on how to do this.

Table 6. Subclass With Replace as Applied to Other Modeled Classes (continued)

Action	Model Option	Discouraged ?	Alternative
	Business Date Field Bytes Message Encoding Character Set Generate Security Is XA Transactional Message Type Queue Connector Factory JNDI Name Reply Queue JNDI Name Response Message Timeout Shadow Type Transactional Transmission Queue JNDI Name	Discouraged	If you feel you have a valid need to change any of these options on application Process, Facade, WSInbound or WebService operations, please raise a Support case.
	Secure Fields	Discouraged	Customers wishing to alter which fields of an application operation are to be treated as Secure should wrap the operation in their own operation, and set the Secure Fields option on the new operation to the desired setting. Affected UIM screen definitions should be repointed at the new operation if required.
Change an operation parameter option	Mandatory Fields	Discouraged	Customers wishing to alter which fields of an application operation are to be treated as Mandatory should wrap the operation in their own operation, and set the Mandatory Fields option on the new operation to the desired setting. Affected UIM screen definitions should be repointed at the new operation if required.

B.4 Subclass Without Replace

B.4.1 Entity

Table 7. Subclass Without Replace as Applied to Entity Classes

Action	Model Option	Discouraged ?	Alternative
Add a stereotyped Entity operation (e.g. <<ns>>, <<nsreadmulti>>, etc.)	None	Allowed	Rather than using Subclass with Replace, add the stereotyped operation through the use of Subclass without Replace. This will ensure that your subclass (and thus your new stereotyped operations) will be treated as 'External', and that you won't get discouraged warnings in Eclipse when you reference them. Note that you will continue to get discouraged warnings if you directly reference operations in the base Entity, as these are Internal - this is by design.

Table 7. Subclass Without Replace as Applied to Entity Classes (continued)

Action	Model Option	Discouraged ?	Alternative
Add a non-stereotyped Entity operation	None	Allowed	Rather than using Subclass with Replace, add the non-stereotyped operation through the use of Subclass without Replace. This will ensure that your subclass (and thus your new non-stereotyped operations) will be treated as 'External', and that you won't get discouraged warnings in Eclipse when you reference them. Note that you will continue to get discouraged warnings if you directly reference operations in the base Entity, as these are Internal - this is by design.
Change the structure of an Entity operation	None	Discouraged	Create a new stereotyped operation using Subclass without Replace.
Change an Entity operation option	Auto ID Field Auto ID Key No Generated SQL Optimistic Locking Order By SQL Where	Discouraged	Create a new stereotyped operation using Subclass without Replace.
	Database Table Level Auditing	Discouraged	This option is settable via runtime properties, if you want to change the behaviour of application operations. Otherwise, create a new stereotyped operation to implement the required functionality using Subclass without Replace.
	On Fail Operation Post Data Access Operation Pre Data Access Operation Treat Readmulti Max as Informational Exception Readmulti Max Records Returned	Discouraged	Use Subclass with Replace to override these options on an application Entity class. Otherwise, create a new stereotyped operation to implement the required functionality using Subclass without Replace.
Change an Entity class option	Enable Validation	Discouraged	Use Subclass With Replace to override this option on an application Entity class. Otherwise, create a new stereotyped operation to implement the required functionality using Subclass without Replace.
	Abstract Allow Optimistic Locking No Generated SQL	Discouraged	Create a new stereotyped operation using Subclass without Replace.
	Audit Fields Last Updated Field	Discouraged	Use Extension Classes to override these options on an application Entity class.

B.4.2 Process, Facade, WebService, WSInbound

Table 8. Subclass Without Replace as Applied to Other Modeled Classes

Action	Model Option	Discouraged ?	Alternative
Change a class option	Abstract Generate FIDs Replace Superclass WS Binding Style WS Is XML Document Document Type Generate Facade Bean Provider Name Request Handlers Response Handlers Validate Request XML Document XML Schema	Discouraged	Create a new operation in a project-specific class, wrapping External APIs of the application class functionality if appropriate, and setting the appropriate options on the new class.
Add an operation	None	Discouraged	Create a new operation in a project-specific class, wrapping External APIs of the application class functionality if appropriate, and setting the appropriate options on the new class operation.
Change an operation	None	Discouraged	Create a new operation in a project-specific class, wrapping External APIs of the application class functionality if appropriate.

Table 8. Subclass Without Replace as Applied to Other Modeled Classes (continued)

Action	**Model Option**	**Discouraged ?**	**Alternative**
Change an operation option	Audit BI Calls Business Date Field Bytes Message Encoding Character Set Generate Security Is XA Transactional Message Type Queue Connector Factory JNDI Name Reply Queue JNDI Name Response Message Timeout Shadow Type Transactional Transmission Queue JNDI Name Secure Fields	Discouraged	Create a new operation in a project-specific class, wrapping External APIs of the application class functionality if appropriate, and setting the appropriate options on the new class operation.
Change an operation parameter option	Mandatory Fields	Discouraged	Create a new operation in a project-specific class, wrapping External APIs of the application class functionality if appropriate, and setting the appropriate options on the new class.

B.5 Domain Overriding

B.5.1 Domain Definitions

Table 9. Overriding Domain Definitions

Action	Model Option	Discouraged ?	Alternative
Change a specific application Domain Definition		Discouraged (partially)	Customization of the following application Domain Definitions is not allowed by customers: TRUNCATED_NOTE_TEXT NOTE_TEXT INCIDENT_DESCRIPTION SERVICE_DELIVERY_NOTE_TEXT_SMALL INJURY_DESCRIPTION ACTION_TAKEN CITIZEN_ACCOUNT_RICH_TEXT CW_RICH_STRING VIEW_LIFE_EVENTS_POST_SUBMIT_XML_DATA RICH_TEXT_EDITOR_WIDGET SEC_RICH_TEXT_VIEW_WIDGET RICH_TEXT PROGRESS_RICH_TEXT_SMALL
Change a Domain Definition option	Codetable Name Codetable Root	Discouraged	Create a new Domain Definition with the appropriate Codetable Name and Root, and wrap in their own processing. Customers are not allowed to change these options for application Domain Definitions.

Table 9. Overriding Domain Definitions (continued)

Action	Model Option	Discouraged ?	Alternative
	Compress Embedded Spaces Convert to Uppercase Custom Validation Function Name Default Maximum Value Minimum Size Minimum Value Pattern Match Remove Leading Spaces Remove Trailing Spaces Storage Type	Allowed	
	Maximum Size	Allowed (Partially)	Allowed for increasing the size only. If you want to decrease the size of an application Domain Definition, please raise a Support case. Not to be used to change the maximum size of the USERNAME Domain Definition.
Change the Type of a Domain Definition	None	Discouraged	Create a new Domain Definition with the appropriate Type, and wrap in your own processing. Customers are not allowed to change the fundamental types of application Domain Definitions.
Change the String Length of a Domain Definition	None	Allowed (Partially)	Allowed for increasing the size only. If you want to decrease the size of an application Domain Definition, please raise a Support case. Not to be used to change the string length of the USERNAME Domain Definition.
Create a new Domain Definition based on an application Domain Definition	None	Allowed	

B.6 Relationships

B.6.1 Assignable

Table 10. Assignable Relationships

Action	Discouraged ?
Make a customer-supplied struct assignable to an application struct or Entity	Allowed
Make an application struct assignable to another application struct or Entity	Discouraged

B.6.2 Aggregation

Table 11. Aggregations

Action	Discouraged ?
Aggregate an application struct in a customer-supplied struct (i.e. create a customer struct that 'contains' an application struct)	Allowed
Aggregate a customer-supplied or application struct in an application struct (i.e. add any struct to an application struct by aggregation)	Discouraged

B.6.3 Foreign Key

Table 12. Foreign Keys

Action	Discouraged ?
Create a new Foreign Key where a customer-supplied Entity is the child	Allowed
Create a new Foreign Key where an application Entity is the child	Discouraged

B.6.4 Index

Table 13. Indexes

Action	Discouraged ?
Create a new Index (on either an application or customer-supplied Entity) using a customer-supplied struct	Allowed
Create a new Index (on either an application or customer-supplied Entity) using an application struct	Discouraged

B.6.5 Unique Index

Table 14. Unique Indexes

Action	Discouraged ?
Create a new Unique Index on an application Entity	Discouraged
Create a new Unique Index on a customer-supplied Entity using an application struct	Discouraged
Create a new Unique Index on a customer-supplied Entity using a customer-supplied struct	Allowed

B.7 Other Mechanisms

B.7.1 Exclusions

Table 15. Exclusions

Action	Discouraged ?
Use Exclusions to attempt to excluded classes from a server build	Discouraged - application classes are not rebuilt every time from version 6.0.3

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785

U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing

Legal and Intellectual Property Law.

IBM Japan Ltd.

19-21, Nihonbashi-Hakozakicho, Chuo-ku

Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Trademarks