IBM Cúram Social Program Management
Version 7.0.0

Cúram Development Compliancy Guide

IBM

Edition

This edition applies to IBM Cúram Social Program Management v7.0.0 and to all subsequent releases unless otherwise indicated in new editions.

Contents

Figures

Tables

Developing Compliantly with Cúram in IBM Cúram Social Program Management, Version 7.0.0

When you develop Cúram applications, you must comply with certain guidelines to ensure that you can easily upgrade to future versions without affecting custom functionality. Complying with these guidelines is also essential to ensure that IBM Support can better support your custom implementation.

There are changes to the guidelines since version 6.0.3. Although all application customization mechanisms continue to be supported for those projects that have already used them, some customization mechanisms are now discouraged for new development.

Server: Changing Source Artifacts

There are many types of server artifacts, some of which are application classes. Some of these artifacts are represented in an application model. Other Java interfaces are "handcrafted". While it is possible to change limited aspects of a modeled interface by changing the model and regenerating code, it is not possible to change a handcrafted interface.

It is important to be able to distinguish between the application implementations of both categories of class.

Modeled interfaces
Appear in the application UML model

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

Some components can contain interfaces that do not fall into either of these categories, and these interfaces are described in component-specific documentation. Both modeled and handcrafted application interfaces can have implementations that can be customized. You must look at an implemented interface to determine its category.

The recommendations on how to change Server Source Artifacts have changed with version 6.0.3. The recommendations in the *Cúram Development Compliancy Guide* are definitive.

Write Source Code for New Methods and Classes

New customer-specific classes are classes that wrap existing classes. They are essentially all new code and 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 dictates the package structure of the new source files.

For handcrafted implementations, it is up to you as to how the new class is packaged. You can use Google Guise to configure new subclasses.

Changing CER Rule Sets

The CER Editor stores its rule sets on the database rather than in the file system. To help identify rule sets that can be customized compliantly, refer to the rule set interdependencies developer documentation that can be found alongside the Data Dictionary. Any rule sets shipped in the core component must not be customized.

The HealthCareRuleSet is documented as customizable, but note that it contains two infrastructure rule classes that must not be customized. They are the AbstractObjective rule class and the DefaultObjective rule class. Similarly, the HealthCareScreeningRuleSet is documented as customizable, but it contains the AbstractProgram rule class, which is considered infrastructure and must not be customized. Solutions might have extra compliancy statements about their rule sets documented elsewhere.

Extending Codetables

Documentation is provided to indicate which codetables are safe to extend, and which codetables require that you contact IBM Support before customizing.

A list of codetables that cannot be extended without contacting IBM Support is provided in the project documentation directory structure for every installation, in a folder called `RestrictedCodeTables`. If you want to customize a codetable that is in this list, raise a PMR on the IBM Support site.

Refer to the procedure Requesting a Product Enhancment (RFE) for a description of how to make the enhancement request.

Starting a New Project

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

Understand the Client and Server 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.

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.
- 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 that are 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 are developed. This convention is not mandatory 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.

Source Code Control

To track all changes to source artifacts, place the development directory structure under source code control. When under source code control, tag all development artifacts.

Ensure that the tag refers to the version of the application. At any point, you can then produce a report by using file comparison tools to identify all files that were added or changed. This report is useful when you are upgrading the application.

From version 6.0.3, changes were made to how Java™ source code is delivered.

Source Code and APIs

All application Java functionality is distributed as previously built JAR files. You must regenerate and rebuild applications in a customer installation only if required by the use of customer extension mechanisms.

The customer build process does not need to rebuild the entire Java source code base; only project-specific source code and any dependent regenerated Java source code needs 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 included as sample code for documentation purposes only. This code is not directly involved in the build process from version 6.0.3 onwards. This sample source code is distributed in JAR files on a per-component basis as follows: `components\<component name>\sample\src.zip` The built versions of each component can be found in the following location: `components\<component name>\lib\<component name>.jar`

Also, from version 6.0.3, class operations are marked as Internal or External by annotations.

External operations are now the official API, which you are encouraged to use and call from your own code.

Important: By default, classes with no annotations are internal.

Internal APIs

Although it is possible to call and subclass Internal APIs from custom code, this practice 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 might be changed or removed in future releases. A minimum notice period of one year is given to customers in respect to 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. That is, there are no current plans to change any of the APIs marked as Internal in 6.0.3. This approach provides adequate time for customers to plan any such migrations.

Existing customer references to APIs that are marked as Internal from version 6.0.3 onwards continue to function as before. However, discouraged warnings are generated within Eclipse projects that have such dependencies.

Try to move your projects away from dependencies on Internal APIs over time, and do not introduce new dependencies on them where possible. Within reason, depending on where a customer project is in its design or development process, it might be inevitable in the short term. Most existing customers will see discouraged references that are reported after they install version 6.0.3 or later versions. It is not expected that customers fix these references immediately as part of an upgrade. These references do not affect their support entitlements.

As with previous versions of the application, some Internal APIs are configured to produce 'access restriction' errors in Eclipse if referenced. These APIs are annotated with `@Accesslevel(RESTRICTED)`. Such references are not supported in customer projects. These APIs were always Internal, and were never supported for customer use. Access-restricted APIs produce Eclipse errors and discouraged APIs produce Eclipse warnings.

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.

The Javadoc for each component can be found at `components\<component name>\doc\api.zip`. Some components might not have any Javadoc if they have no External APIs. Only reference classes that are documented in Javadoc from customer code; referencing other classes produce discouraged warnings or access restricted errors and are not supported.

As with all APIs, it is expected that classes that are marked as External will evolve over time, while remaining compatible with previous versions. If you require some capability that cannot be fulfilled through a combination of External APIs and allowed extension mechanisms, raise your requirements through Support. If appropriate, a new API, customization hook, strategy pattern or configuration-based approach is made available, and such new APIs can be delivered in Feature Packs. Alternatively, an existing Internal API might, in some circumstances, be re-designated as External, if appropriate.

Server: Extension Mechanisms

The removal of source code from the areas of key functionality has resulted in a change to the recommended approach to using extension mechanisms on customer projects.

Previously, if customers wanted to use the various application extension mechanisms (for example, extension classes, subclass with and without replace, aggregation), they might search across the codebase to see where and how target classes were being invoked within application code. They might then make an assessment of the functional effects of the extension that is being considered.

Customers do not have the source code for some areas of key functionality, and in addition a large number of APIs are marked as Internal. The following section summarizes the 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 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 are provided in "Server: Discouraged Extension Mechanisms" on page 12. Where mechanisms are discouraged, appropriate alternative mechanisms to be employed by customers are listed.

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 are marked as External, and can be used by customers.

However, the addition of stereotyped and non-stereotyped operations to application Entities is still allowed, as is the setting of a number of Entity options.

Before version 6.0.3, attributes could be added to application Entity classes using extension. However, because source code is being removed for areas of key functionality, customers no longer have visibility whether or not the attributes that are added by extension classes are mapped to external APIs. For this reason, adding attributes to application Entity classes is now discouraged.

Customers that want 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, because this is required by the Evidence Generator.

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

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

Domain Definitions

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

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 (because they no longer have full visibility on where these application structs are being used). However, customers can continue to use aggregation to include application structs in their own project-specific structs.

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.

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

Similar to 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 pointed at the new Facade operations if you want.

Non-Modeled Classes

Some components contain non-modeled classes. For these classes, the use of each External interface or class is described in the Javadoc information 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 because it might 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 to call or reference package protected classes.

Many non-modeled APIs are not directly customizable. Only interfaces or classes that are tagged with the `@Implementable` annotation can be extended or implemented. Refer to Javadoc information detailing how to customize or implement such classes. Non-modeled classes that are not tagged with the `@Implementable` annotation must not be extended or implemented because new operations might be added over time, which might 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. Strategy classes are typically named 'xxxStrategy', so they can be easily identified.

Related concepts:

"Component Compliance Details" on page 10
Read the following compliance information for individual components.

Summary Guidance

Summary guidance for referencing or customizing application classes.

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. Do 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, follow the detailed guidance for allowed customization.
- If the class is Non-Modeled, refer to its Javadoc or any configuration/ development guide for its parent component for details of customization points.

Avoiding Common Compliancy Pitfalls

Read about compliancy issues that can arise and guidelines for avoiding these issues. Following these guidelines from the early stages of a project is relatively easy. However, if you do not follow the guidelines, it can result in serious disruptions later and fixing these disruptions can be both costly and difficult.

Use Project-specific Prefixes in Artifact Names

Prefix all new source artifact names with a relevant acronym or abbreviated word. By using a project-specific prefix, you can prevent naming collisions from occurring between new artifacts that you add, and new artifacts that IBM might add over time. Naming collisions can be difficult to fix when they are identified after the fact.

Use the same acronym or abbreviated word throughout. As the project progresses, this prefix makes project additions to core artifacts more obvious. This distinction becomes more useful as the development effort grows. Most projects are described by some kind of acronym and this acronym is a good candidate to use as the prefix.

Some artifact types have more than one identifier and you must remember this fact when you are naming them.

It is important to note that the use of project-specific prefixes might not apply when overriding some application artifacts. Where supported, override mechanisms typically require the custom artifacts to have the same name as the default artifacts that they override, but there are exceptions.

There are further considerations as follows:

- There are many different types of identifiers. For example, a file name, an XML ID, a Java class name, or a combination of identifiers.
- A short prefix is advisable because there might be restrictions on name lengths. For example, some types of database identifiers have length restrictions.

Note: In addition to source artifacts, it is also important to consider identifier values that might conflict with values used by IBM.

Use project-specific prefixes in custom artifact names

Avoid naming collisions when you take on new versions of the software by ensuring that you always name new, custom artifacts with a consistent prefix for your project.

Some artifact types have more than one identifier. Remember this when you name your custom artifacts. The following list describes examples of common development artifacts that can cause naming collisions when you take on a new release.

Database fields
New database fields can be delivered in fix packs. Use project prefixes for database fields to prevent duplicating the names delivered in the fix packs.

Application code table items
New application code table items can be delivered in fix packs. Use a project prefix when you name custom code table items to prevent duplicating the names delivered in the fix packs. Custom code table items have a value and a Java identifier, and both share a flat namespace with application items in the same code table.

Entity classes

Custom Entity classes have a table name that shares the flat namespace and database schema with application tables and must have a unique table name within that namespace. It also has a Java class name, which shares a hierarchical namespace and package structure with application Java classes. Use project-specific prefixes for custom entity classes to prevent duplication of the names of the new entity classes.

Identifier values that conflict with values used by IBM
Consider identifier values that might conflict with values used by IBM. For example, the TransactionInfo.setFacadeScopeObject and TransactionInfo.getFacadeScopeObject APIs enable developers to access objects that are associated with the current transaction. When you use these APIs, use a String as your object identifier and prefix this string with an appropriate project-specific word to ensure that your data for the transaction does not conflict with IBM data.

Use Numeric Identifiers in Custom Initial and Demo Data

Pre-defined initial and demo data is loaded into an application database using DMX files. This data is installed into the database when a system is first set up, or when a system is upgraded. You might also want or need to add your own initial or demo data or both.

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 (for example, primary keys) for your initial and demo data are drawn from reserved ranges. Therefore, a set of ranges has been reserved for customer use.

Reserved ranges for unique identifiers for primary keys

Projects need to use unique identifiers (primary keys) in their custom initial and demonstration data that are drawn from reserved ranges.

The following are the reserved ranges:
- Non-human readable primary keys: 45,000 - 49,999 (inclusive)
- Human readable primary keys: 11,521 - 12,799 (inclusive)
- Rule sets: 4,500 - 4,999 (inclusive)

Large Data Sets

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

Instead of using keys from the allocated ranges, use the key server to generate the key values required. If this data is to 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 about this process, contact IBM Support for further information.

Refer also to the procedure Requesting a Product Enhancment (RFE) for a description of how to make the enhancement request.

Codetables Exception

Note that the Large Data Sets statement does not apply to code tables.

Avoid In-Place Modifications to Application Files

Fix Pack and iFix releases must be able to safely move, restructure, or overwrite application files. If shipped product files are modified, upgrades might overwrite them without notice and changes might not be compatible with the modifications. Reapplying the in-place changes afterward might not be possible.

Client and Server: Exceptions for In-Place Modifications

A list of the small number of exceptions to the in-place modifications rule for Client and Server development.

- 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`

Never Create Dependencies on Sample or Demo Artifacts

Never create dependencies on Sample or Demo artifacts. Never rely on dependencies or references to sample or demo artifacts from custom code. Sample or Demo artifacts are subject to change without notice

Different product areas in Cúram take different approaches to marking artifacts as Internal, Sample, or Demo, so this information cannot give a concise statement of how to identify them. However, there are a few instances where they can be identified. These instances are 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 IBM Support.

Refer also to the procedure Requesting a Product Enhancment (RFE) for a description of how to make the enhancement request.

The CPMSample folder

The `CPMSample` folder is internal; all code and artifacts within this folder can change without any notice. If customers want to use functionality in the `CPMSample` folder, they must duplicate it in their code base.

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 (for example, code tables and workflows). 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 Fix Packs or other upgrades. In every case, there must be one central place where concurrent changes are merged and validated and this is the development codebase. The system of record for these artifacts is the development codebase.

Do not Create New Dependencies on Internal APIs

From version 6.0.3 onwards, avoid calling or customizing application classes and operations that are marked as Internal, as such APIs might change in subsequent versions of the application.

Component Compliance Details

Read the following compliance information for individual components.

Important: Unless otherwise indicated, for all components (whether listed here or not) it can be assumed that the 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, see the *Cúram Web Client Reference Manual*.Note that files from the `CuramCDEJ` folder are 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 produces discouraged warnings or access restricted errors and are not supported. **Important:** 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 that are used to access the application database. If necessary, these drivers can 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 was not shipped in the product, that is, was not tested and verified for use with the application, the customer might be requested to replace the driver with a tested version, while the specific issue is raised with the third-party vendor.Note that files from the `CuramSDEJ` folder are 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 subpackages). For more information about how to use these APIs, see the *Persistence Cookbook*.
CER Infrastructure	The compliancy statement for CER Infrastructure can be found in the *Cúram Express Rules Reference Manual*. CER entities, that is, any entity whose name is prefixed by the word Creole, are considered Internal and subject to change, and customers should not update them or query them except through the CER API or DMX files.
Dependency Manager	The Dependency Manager encompasses all server artifacts in the `curam.dependency` code package and all its subpackages. 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 development 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, see the *Inside Cúram Eligibility and Entitlement Using Cúram Express Rules Guide*.
Evidence Generator	The Evidence Generator is application infrastructure that is included as part of the Tools directory structure (EGTools). For more information about using the Evidence Generator, see the *Cúram Evidence Generator Specification*.
DocMaker	No part of the DocMaker tool might be customized.

Table 1. Component Compliance Details (continued).

This table lists the components with Non-Modeled APIs.

Component	Details
Pod Infrastructure	Pod Infrastructure is included in the `widget-inf.jar` and `widget-utility.jar` files. The Pod Infrastructure cannot be customized. Pod-Loaders cannot be customized. For more information about developing Pods, see the *Cúram Pod Developer's Guide*.
Funded Program Management	For guidelines on how to customize this component, 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, see the *Persistence Cookbook* and the component Javadoc.
Cúram Citizen Context Viewer	For further guidelines on how to customize this component, see the *Cúram Citizen Context Viewer Configuration Guide* and the component 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, see Part VI of the *Cúram Workflow Reference Guide*.
Cúram Waitlists	For guidelines on how to customize this component, see the *Cúram Waitlist Customization Guide* and the component Javadoc.
IBM Cúram Business Intelligence and Analytics	For guidelines on how to customize this component, see specific compliancy guidelines for Business Intelligence and *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, see the *Cúram Universal Access Developers Guide* and the component 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, see the *Cúram Provider Management Developer Guide* and the component Javadoc.
IBM Cúram Youth Services(CYS)	For guidelines on how to customize any CYS Entities or replacing any CYS implementation, see the *Persistence Cookbook* and the component Javadoc.
IBM Cúram Child Care (CCC)	For guidelines on how to customize any CCC Entities or replacing any CCC implementation, see the *Persistence Cookbook* and the component Javadoc.

Server: Discouraged Extension Mechanisms

Many of the extension mechanisms that were recommended in versions before 6.0.3 to extend or replace application classes are now discouraged. Read about which mechanisms are allowed and which are discouraged when applied to which class types, and what to do if you find that a mechanism/class type combination that you want to employ is now discouraged.

Extension Classes

Discouraged extension mechanisms for extension classes.

Entity

Extension classes as applied to entity classes.

Table 2. Extension Classes as Applied to Entity Classes

Action	Model Option	Discouraged?	Alternative
Add a stereotyped entity Operation (for example, `<<ns>>`, `<<nsreadmulti>>`)	None	Discouraged	Rather than using an `<<extension>>` class, use subclass without replace to add the stereotyped operation.
Change an entity Operation (or example, parameters)	None	Discouraged	Use subclass without replace to create a new stereotyped operation with the structure you want. If you feel you have a valid need to change the structure of an application Entity operation, raise a Support case. Refer to the procedure Requesting a Product Enhancement (RFE) for a description of how to make the enhancement request.
Change an Entity operation option	Auto ID Field Auto ID Key No Generated SQL Optimistic Locking Order By SQL Where	Discouraged	Use subclass without replace to create a new stereotyped operation. If you feel you have a valid need to change these options on application Entity operations, raise a Support case.
	Database Table-level Auditing	Discouraged	Use runtime properties to set this option.
	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 only to change these options on application Entity operations.
Change an Entity class option	Enable Validation	Discouraged	Use Subclass with Replace only to change this option on application Entity operations.
	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, raise a Support case.

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

Action	Model Option	Discouraged?	Alternative
	Audit Fields Last Updated Field	Allowed	Currently only Discouraged when you are using Extension classes, and it continues to be Discouraged from 6.0.3
Add an Entity attribute	None	Discouraged	If you want to add data to application screens, add new customer-specific Entity classes, and wrap Cúram CRUD operations in your own process classes to maintain both tables atomically. Cúram screens can then be changed to point to the new process class.
Change an Entity attribute option	Allow Nulls	Discouraged	If you feel you have a valid need to change this option on application Entity attributes, raise a Support case.

Struct

Extension classes as applied to struct classes.

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, raise a Support case. Refer to the procedure Requesting a Product Enhancement (RFE) for a description of how to make the enhancement request.

Table 3. Extension Classes as Applied to Struct Classes (continued)

Action	Model Option	Discouraged?	Alternative
Change a struct option	Audit Fields	Discouraged	If you need to propagate Audit Fields from an Entity to a screen: • Create new stereotyped operations to maintain the Audit Fields, • cCreate a new Facade that wraps the existing Entity CRUD operations and calls the new stereotyped operations • Update any UIM pages as necessary

Process, Facade, WebService, WSInbound

Extension classes as applied to other modeled classes.

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, raise a Support case. Refer to the procedure Requesting a Product Enhancement (RFE) for a description of how to make the enhancement request.
Add an operation	None	Discouraged	Using extension classes to add an operation was never encouraged because customers would be required to modify the application Java code in-place. 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.

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

Action	**Model Option**	**Discouraged?**	**Alternative**
Change an operation (for example, operation visibility)	None	Discouraged	Create an operation in a project-specific class and wrap the 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, raise a Support case.
Change an operation option	Audit BI Calls	Discouraged	Use runtime properties to set this option.
	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, raise a Support case.
	Secure Fields	Discouraged	To alter which fields of an application operation are to be treated as Secure, wrap the operation in your own Facade class and operation, and set new options for the Secure Fields. Redirect the affected UIM screen definitions to the new operation, if necessary.
Change an operation parameter option	Mandatory Fields	Discouraged	To alter which fields of an application operation are to be treated as Mandatory, wrap the operation in your own Facade class and operation, and set new options for the Mandatory Fields. Redirect the affected UIM screen definitions to the new operation, if necessary.

Subclass With Replace

Discouraged extension mechanisms for Subclass With Replace.

Entity

Subclass With Replace as Applied to Entity Classes.

Table 5. Subclass With Replace as Applied to Entity Classes

Action	Model Option	Discouraged?	Alternative
Add a stereotyped Entity operation (for example, `<<ns>>`, `<<nsreadmulti>>`)	None	Discouraged	Use Subclass without Replace to add the stereotyped operation. Using Subclass without Replace ensures that your subclass and your new stereotyped operations are treated as 'External', and you will not receive discouraged warnings in Eclipse when you reference them. **Note:** You continue to get discouraged warnings if you directly reference stereotyped operations in the base Entity because, by design, they are Internal.
Add or Change a non-stereotyped Entity operation	None	Discouraged	Use Subclass without Replace to add a non-stereotyped operation. Using Subclass without Replace ensures that your subclass and your new non-stereotyped operations are treated as 'External', and that you do not receive any discouraged warnings in Eclipse. **Note:** You continue to get discouraged warnings if you directly reference operations in the base Entity because, by design, they are Internal. Customers are discouraged from providing new implementations for non-stereotyped application Entity operations.
Change the structure of an Entity operation	None	Discouraged	Use Subclass without Replace to create a new stereotyped operation. If you feel you have a valid need to change the structure of an application Entity operation, raise a Support case. Refer to the procedure Requesting a Product Enhancement (RFE) for a description of how to make the enhancement request.

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

Action	Model Option	Discouraged?	Alternative
Change an Entity operation option	Auto ID Field Auto ID Key No Generated SQL Optimistic Locking Order By SQL Where	Discouraged	Use Subclass without Replace to create a new stereotyped operation. If you feel you have a valid need to change these options on application Entity operations, raise a Support case.
	Database Table Level Auditing	Discouraged	Use runtime properties to set this option.
	On Fail Operation Post Data Access Operation Pre Data Access Operation	Allowed (Partially)	You are still allowed to implement application Entity exit points. If you want to process in exit points for which there is a default implementation, the default implementation must be called at the beginning of the customer exit point implementation (that is, ensure that a call to 'super()' at the appears at the beginning). Customers are not allowed to turn 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)	Customers are still allowed to implement application Entity exit points. If you want to process in exit points for which there is a default implementation, the default implementation must be called at the beginning of the customer exit point implementation (that is, ensure that a call to 'super()' at the appears 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, raise a Support case.
	Audit Fields Last Updated Field	Discouraged	Use Extension classes to override these options on an application Entity class.

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

Action	Model Option	Discouraged?	Alternative
	Replace Superclass	Allowed (Partially)	Implicitly allowed to support other 'Allowed' actions that are described in this table.

Process, Facade, WebService, WSInbound

Subclass With Replace as Applied to Other Modeled Classes.

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 an 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, raise a Support case. Refer to the procedure Requesting a Product Enhancment (RFE) for a description of how to make the enhancement request.
Add an operation	None	Discouraged	Create an operation in a project-specific class, wrapping External APIs of the application class functionality if appropriate.
Change an operation	None	Discouraged	Create an 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, raise a Support case.
Change an operation option	Audit BI Calls	Discouraged	Use runtime properties to set this option.

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, raise a Support case.
	Secure Fields	Discouraged	If you want to alter which fields of an application operation are to be treated as Secure, wrap the operation in its own operation, and set the Secure Fields option to the correct setting. Redirect the Affected UIM screen definitions to the new operation, if required.
Change an operation parameter option	Mandatory Fields	Discouraged	If you want to alter which fields of an application operation are to be treated as Mandatory, wrap the operation in its own operation, and set the Mandatory Fields option to the correct setting. Redirect the Affected UIM screen definitions to the new operation, if required.

Subclass Without Replace

Discouraged extension mechanisms for Subclass Without Replace.

Entity

Subclass Without Replace as Applied to Entity Classes.

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.
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.

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

Action	Model Option	Discouraged?	Alternative
	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.

Process, Facade, WebService, WSInbound

Subclass Without Replace as Applied to Other Modeled Classes.

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 an operation in a project-specific class, wrap External APIs of the application class functionality, if appropriate, and set the appropriate options on the new class.
Add an operation	None	Discouraged	Create an operation in a project-specific class, wrap External APIs of the application class functionality, if appropriate, and set the appropriate options on the new class operation.
Change an operation	None	Discouraged	Create an operation in a project-specific class and wrap 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 an operation in a project-specific class, wrap External APIs of the application class functionality, if appropriate, and set the appropriate options on the new class operation.
Change an operation parameter option	Mandatory Fields	Discouraged	Create an operation in a project-specific class, wrap External APIs of the application class functionality, if appropriate, and set the appropriate options on the new class.

Domain Overriding

Discouraged extension mechanisms for Domain Overriding.

Domain Definitions

Overriding 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 Domain Definition with the appropriate Codetable Name and Root, and wrap it in your own processing. Customers are not allowed to change these options for application Domain Definitions.
	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	

Table 9. Overriding Domain Definitions (continued)

Action	Model Option	Discouraged?	Alternative
	Maximum Size	Allowed (Partially)	Allowed for increasing the size only. If you want to decrease the size of an application Domain Definition, 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 Domain Definition with the appropriate Type, and wrap it 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, raise a Support case. Refer to the procedure Requesting a Product Enhancement (RFE) for a description of how to make the enhancement request. Not to be used to change the string length of the USERNAME Domain Definition.
Create a Domain Definition based on an application Domain Definition	None	Allowed	

Relationships

Discouraged extension mechanisms for Relationships.

Assignable

Discouraged extension mechanisms for Assignable Relationships.

Table 10. Assignable Relationships

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

Aggregation

Discouraged extension mechanisms for Aggregations.

Table 11. Aggregations

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

Foreign Key

Discouraged extension mechanisms for Foreign Keys.

Table 12. Foreign Keys

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

Index

Discouraged extension mechanisms for Indexes.

Table 13. Indexes

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

Unique Index

Discouraged extension mechanisms for Unique Indexes.

Table 14. Unique Indexes

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

Other Mechanisms

Discouraged extension mechanisms for other mechanisms.

Exclusions

Discouraged extension mechanisms for Exclusions.

Table 15. Exclusions

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

Patched Files

Discouraged extension mechanisms for Patched Files.

Table 16. Patched Files

Action	**Discouraged?**
Substituting patched files for out-of-the-box files.	Discouraged

Notices

Privacy Policy considerations

can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies or other similar technologies that collect each user's name, user name, password, and/or other personally identifiable information for purposes of session management, authentication, enhanced user usability, single sign-on configuration and/or other usage tracking and/or functional purposes. These cookies or other similar technologies cannot be disabled.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at http://www.ibm.com/privacy and IBM's Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at http://www.ibm.com/software/info/product-privacy.

Trademarks

Printed in USA