Merative Social Program Management 8.1

System Administration Guide

Note

Before using this information and the product it supports, read the information in Notices on page 45

Edition

This edition applies to Merative™ Social Program Management 8.0.0, 8.0.1, 8.0.2, 8.0.3, and 8.1.

© Merative US L.P. 2012, 2023

Merative and the Merative Logo are trademarks of Merative US L.P. in the United States and other countries.

Contents

1 Configuring the system

Configure the functions of the application. For example, configure the following: application properties, case audit selection queries, communication templates, batch processes, security settings, business intelligence reports, target systems, content management interoperability services.

System administration includes functionality for managing a broad spectrum of elements that impact the operation of the application. System administration requires some familiarity with technical terms, as certain components of system administration can only be created during application development. For example, the execution of batch processes is requested from the system administration module; however, the batch processes themselves can only be designed and implemented as part of application development.

Other application components can be maintained as part of system administration, but must still be referenced in the application as part of application development. This includes code tables and rate tables.

1.1 Configuring the Rich Text Editor to filter corrupted content

Use a hook point to filter already corrupted characters that might be present in the Merative™ Social Program Management database.

Users can enter, format, and style text by using the Social Program Management Rich Text Editor. Using emojis in notes and characters, for example É, ï, &, can corrupt characters in the database or in an existing Social Program Management system.

Where emojis or extended characters cause an issue with the display of notes, you can filter the emojis or extended characters by using a hook method.

Use the `curam.core.hook.impl.NoteFilterHook.preFilterNoteContent` hook to filter the contents of a note before the note is displayed in the application.

By default, the interface NoteFilterHook is implemented by `curam.core.hook.impl.NoteFilterHookImpl.preFilterNoteContent`.

The hook point is called to process rich text notes for all the functions that launch `curam.core.sl.entity.impl.Note#getLastNoteText1`.

> **Note:** The hook point is called to process rich text notes for display purposes only.

The following list outlines the methods that call the filter that is provided through the hook point:

- `curam.core.sl.impl.UserAccess#getNote1(CaseNoteKeyStruct)`
- `curam.core.sl.impl.ParticipantNote#read1(ParticipantNoteKey)`
- `curam.core.sl.impl.ContactLog#determineNoteText(ContactLogKey, NoteKey)`
- `curam.piwrapper.impl.NoteImpl#getNotesText()`

Organizations can enable the hook point through the standard Guice dependency injection mechanism.

1.2 Configuring the application

Configure the configuration options specific to the running of the runtime application, for example property configuration, code tables, and language locale configuration.

Configuring application properties

Application properties configure parts of the runtime application, they allow the system administrator to customize the application to suit the organization's needs without having to build and redeploy the application.

Application properties are variables that are used by the system in a different ways; for example, some properties alter the functionality provided by the system and therefore allow the system to be configured to suit an organization's needs. The values for these variables can be maintained at runtime, thereby providing a mechanism for changing functionality dynamically without the necessity of going through a full development cycle to implement changes. An example of such a variable is the property that denotes the default date format used by the application, *curam.misc.app.defaultdateformat*. The value of this property can be *'Date_mdy_ext'* or it can be changed to *'Date_dmy_ext'*.

Browsing for a property

Properties can be browsed for and filtered by locale and category. Property categories are broadly divided between Application and Infrastructure categories. Categories group similar property types together in order to simplify the management of related property types. For example, one type of property category is 'Application - Address Settings'. This property category contains all the properties that relate to address settings in the application.

Adding a property to the application

Properties can be added to the application. Information maintained for each property includes the locale, current value, default value, and the category of the property.

The default value specifies the value that the application property will be reset to if a user resets the property defaults for the application. The locale is primarily used to distinguish the language for the property description and display name. For example, *en-US* (US English). The display name is the property name displayed to a user. For example, the e-mail server property that is used by the application would have a display name, *curam.notification.notificationemailserver*. The description provides more detailed information about the functionality of the property. Both the display name and the description should be written in the language described in the locale setting.

Creating a property description

Property descriptions provide multi-language descriptions for application properties. A property description includes the locale, display name, and the description for the property. Using multi-language property descriptions ensures that users in multiple locales will be able to understand the application properties. Only one property description can be entered for each locale.

Publishing property changes

Changes made to properties are not propagated to the application until the changes are published. Application properties have a dynamic setting, which determines whether published changes to the application property dynamically affect the system. If a property is defined as static, then published changes to this property do not take effect until the system is rebooted. Static property changes do not take effect until reboot because they contain information that cannot be updated while the application is running. An example of a static property would be 'curam.db.type=DB2', which indicates a connection to a IBM® Db2® database. This connection cannot be broken while the application is running. Thus, if the value is changed to 'curam.db.type=ORACLE', which indicates a connection to a Oracle™ database, this change cannot be implemented until the server is rebooted.

Resetting to default properties

Application properties can be reset to their default property values. Application properties that have a dynamic setting are changed immediately. Static properties are reset to their default value after the server is rebooted.

Configuring code tables

A code table is made up of a number of code table items, each code table item represents a selection in a drop-down field. Most of code table information is contained within code table items. A code table item contains the actual code that is stored in the application database when that code table item is selected in a drop-down field in the runtime application.

Code tables save space in the application database. By storing drop-down field selections as codes rather than the full text of the selection, you can save database space. For example, rather than storing the ethnicity 'Native American' or 'Alaskan Native' in the database, the application can store the code 'ETH4'. Code tables also allows for the localization of drop-down fields. Localization allows drop-down fields to contain values appropriate to a user's language and dialect.

Code table items also contain a description, which is the text that is displayed in a drop-down field, and the language setting, which contains information relating to the localization of the code table item. A code table has a default code table item. This is the code table item that a drop-down field defaults to.

Adding code tables

Code tables can be added to the application. A unique name must be entered. Once named, code table items can then be added to the table. The order in which the code table items are displayed can be specified. Code table items can be set as selectable. If the selectable indicator is set, the code table item will appear in the drop-down field populated by the parent code table. A language locale can also be set for the code table items. Locales identify a specific language and geographic region. The localization of the application is supported in a number of different languages. Each supported language is specified by a language-locale mapping. For example, English is mapped to the en locale.

Changes made to code tables or code table items are not propagated to the application's drop-down fields until the changes are published (or until the application server is rebooted).

Localizing code tables

Code table drop-down fields can be localized. Localized drop-down fields contain values appropriate to a user's language and country. The combination of language and country is known as a *locale*. Examples of locales include en-US (US English), en-GB (UK English), and es-US (US Spanish).

By using locales, the application can display different drop-down lists for users with different locales. For example, if a user's locale is set to US-Spanish, a drop-down field for the days of the week can display the values Lunes, Martes, Jueves, etc. Conversely, if another user's locale is set to US-English, then the same drop-down field can display the values Monday, Tuesday, Wednesday, etc.

Two code table item settings apply to drop-down field localization. The first is the description setting. This is the text that a user sees in a drop-down field. The second setting is language; this setting refers to the locale of the code table item. Note that although this setting is named language to make it more understandable, it actually refers to a locale, which contains both language and country information.

In multi-locale environments, a locale-specific version of each code table item should be recorded for all code tables. For example, imagine a code table that described the days of the week in an environment where both English and Spanish are used. This code table would require two code table items with a code value of DAY1 - an English code table item with a description of Monday, and a Spanish code table item with a description of Lunes. Having a language-specific code for each day of the week ensures that users are presented with all days of the week, no matter what language is displayed by the application.

Code table hierarchies

Code tables can be used to group other code tables in a hierarchy. Any number of code tables can be included in a code table hierarchy. A code table hierarchy allows the values available for selection in the drop-down field for one code table to be determined by the value selected in the drop-down field for another code table. For example, the values available when selecting the type of special caution to be recorded for a participant can be derived from the category of special caution selected. Code table hierarchies can be viewed and modified from the system administration application. For more information on code table hierarchies, please consult the *Server Developers Guide*.

Working with configurable validations

Validations control the data that is entered by users, for example, to enforce data integrity or to prevent the entry of inconsistent data.

An example of a validation is "The Case Participant Role start date must not be later than the Case end date - "%1d".". This is displayed when a user attempts to add a case member to a case with a start date later than the end date of the case.

Although all validations included in the application are run during processing by default, some validations are verified to not be required as they have no impact on system processing if they are not run. These validations are pre-defined as configurable validations and can be disabled by an agency.

The system administration application provides an agency with the ability to search for and disable validations that are identified as configurable validations. All other validations cannot be maintained within the application and are run during processing.

An organization might have a requirement to disable a validation that is not identified as a configurable validation. The organization can create a produce enhancement request for the validation to be made configurable at the [Merative ™ Ideas Portal](). The validation is then analyzed and it is determined whether it can be reclassified as a configurable validation.

Enabling and disabling configurable validations

Enable and disable configurable validations.

The unique validation reference for a validation, for example `bpocaseparticipantrole. err_caseparticipantrole_xfv_from_date_caseheader_end_date|a|,` searches for and retrieve a validation. The validation reference for a validation can be identified using the Validation Messages HTML Documentation as described later in this section.

So the application can uniquely distinguish validations, the validation reference for each configurable validation is made up of a combination of the message catalog ID and an alphabetic constant that is used to distinguish between the validations used across different modules. In addition, the validation reference for any validations that use the same message text as other validations within a module will also have a numeric constant appended at the end. These constants are arbitrarily assigned with no specific meaning to the letters or numbers used.

The search available in the system administration application only returns validations that are identified as configurable validations and are an exact match to the unique validation reference entered.

An administrator can view whether the configurable validation is currently enabled or disabled, and can select to disable or enable the validation as required. Disabled validations are executed during system processing.

Administrators can also view a list of all validations that are currently disabled by leaving the search field blank.

Using the validations messages documentation

To determine what validations are identified as configurable validations that can be disabled, refer to the Validation Messages documentation that is provided in HTML format as part of the development installers.

Validation documentation

When you run the development installer, the Validation Messages documentation is located in the top level *Doc* folder in the installed code base. The home page of the Validation Messages documentation is accessed by selecting the index.html file located in the *ValidationMessages/html* folder. View the list of the validations referenced by each method of a facade class by selecting the **Facades A-Z Index** link from the home page of the documentation, selecting a **Facade Class Name**, and then selecting a method. View a list of the validations referenced by each method of a screen by selecting the **Screens A-Z Index** link from the home page of the documentation, selecting a Screen Name, and then selecting a method.

For each validation, the validation reference and validation message text are provided. For example, for the validation used to prevent the start date of a case participant role from being later that the end date of a case, the validation reference of `BPOCASEPARTICIPANTROLE.ERR_`

`CASEPARTICIPANTROLE_XFV_FROM_DATE_CASEHEADER_END_DATE` and validation message text of

```
"The Case Participant Role start date must not be later than the
 Case end date - "%1d"."
```

are displayed.

For each validation, the documentation also displays whether the validation is configurable. All configurable validations are verified to have no impact on system processing if they are not executed, and can be disabled by an agency. If the validation is not indicated to be a configurable validation, then the organization cannot disable the validation within the system administration application, and should raise a support case if there is a requirement to be able to disable the validation.

Configurable messages

The Configurable Messages section lists all validations that are identified as configurable validations, including the facades that the validations are referenced from. Access this section by selecting the **Configurable Messages A-Z Index** link from the home page of the documentation. Use this list to determine the unique validation reference that is required to search for and disable the validation as part of system administration. For example, if the organization wishes to identify the validations that exist and are configurable as part of the modify case member process, it can select the "Screens A-Z Index" option from the home page of the documentation, enter "modifyCaseMember" to filter the list of screens, select the "Case_modifyCaseMemberFromList.uim" screen, select the "Case.modifyCaseMember" facade reference, and then view a list of validations, including whether or not each validation is a configurable validation.

If the organization finds that the *BPOCASEPARTICIPANTROLE.ERR_ CASEPARTICIPANTROLE_XFV_FROM_DATE_CASEHEADER_FROM_DATE* validation is configurable, it can then cross reference the validation with the list of configurable validations to identify the unique validation reference ID for the validation. This is done by using the Facade Reference and the Message Reference filters. If the *modifyCaseMember* method name is entered in the Facade Reference filter and the *BPOCASEPARTICIPANTROLE.ERR_ CASEPARTICIPANTROLE_XFV_FROM_DATE_CASEHEADER_FROM_DATE* validation reference is entered in the Message Reference filter, a unique validation reference of *bpocaseparticipantrole. err_caseparticipantrole_xfv_from_date_caseheader_from_date|a|* is displayed for the validation.

Unique validation references

If more than one unique validation reference is displayed for a particular Facade Reference and Message Reference combination, the organization can determine which unique validation reference to disable by enabling the display of the unique validation reference within the application alongside the validation message text that is displayed when business processing is executed that results in the validation being called. This will in most cases result in the display of only one validation reference, which is the validation that should be disabled. In the unlikely event that more than one validation reference is displayed when the business process is executed, this represents a situation in which the same validation is being performed twice during the same business process and both should be disabled.

Additional information

All message catalog entries that are referenced by a method are displayed for each method in the Validation Messages documentation, therefore other types of messages such as infrastructure messages and messages used for logging may also be displayed along with the validation messages. For each method the documentation also displays all messages that are referenced by any method called by the method, and so in some cases, messages are listed that are not necessarily called by the screen that uses the method.

Enabling the display of the unique validation reference

Identify the unique validation reference for a validation by enabling the application property *curam.validationmanager.displayreference.enabled*.

curam.validationmanager.displayreference.enabled enables the display of the unique validation reference beside the validation message text displayed in the application. The unique validation reference can then be used to search for and disable or enable the validation if it is a configurable validation. In some circumstances, such as for validations that are not controlled by the validation manager, no validation reference is displayed even when the application property is enabled. Any such validations are not configurable validations.

8.1.0.0 Configuring text translations

In multi-locale environments, a translate text option is available to enable you to add language-specific translations for text fields such as the name of a quick link that are configured in the administration application and displayed to the user in the caseworker application. Adding a text translation for a field allows users to view the information in the same language in which they are viewing the Social Program Management application.

You can configure text translations in the administration application. You create and maintain a text translation for a business object field by recording the language type for the text translation and the translated text. When text translations exist, Social Program Management reads the locale of the logged in user, matches it with the appropriate translation, and displays the text in the language based on the user's locale.

Configuring language and locale mappings

Language and locale mappings are used to customize the user interface language. They are critical to many culturally and linguistically sensitive data operations, for example, locale information is used when generating pro forma communications.

Each language has a single locale associated with it. The choice of languages available for creating a new language locale mapping is itself populated from the list of languages that are available on the system.

8.0.3.0 Configuring autorecovery

Autorecovery protects users from losing in-progress work due to a system interruption. When caseworkers log back in to the application after an interruption, autorecovery restores the user's Social Program Management tab session and the previously open modal window with the data that was last entered. This means that no data is lost and users can continue from where they stopped before the interruption.

Enabling data autorecovery

Autorecovery is a global setting that applies to all application users and to all areas of the application. There are three properties that relate to autorecovery that organizations can configure through the system administrator application. Organizations can enable autorecovery so that caseworkers' data is saved when a system interruption occurs. Organizations can also configure the length of time between the user system interaction and backups.

> **Note:** Organizations must enable autorecovery. By default, autorecovery is not enabled.

> **Note:**
>
> For optimum use, enable autorecovery during system downtime and not during runtime. For autorecovery to take effect, a server restart is required.

Table 1: Autorecovery system configuration properties

Property	Display name	Default value	Description
Enable autorecovery	`curam.sessionmanagement.t`	false	Enable the property so that the user's data can be autorecovered because of a system interruption. The setting applies to all the application areas where autorecovery is supported. If the value is set to true, the system recovers any unsaved data that caseworkers entered before the interruption occurred and returns the caseworker to where they were in the application before the interruption. If the value is set to false, the system does not recover unsaved data in any part of the application. The property is dependent on the Curam FormsAPI.
Enable Curam Forms API	`curam.sessionmanagement.t`	false	The CuramFormsAPI exposes a programmable interface for forms in modals. Organizations can use the API to listen for changes in the form fields, get the field value and set the field value. To enable the CuramFormsAPI, set the property to true. > **Note:** The CuramFormsAPI is supported only for use with the autorecovery feature.
Autorecovery throttle interval property	`curam.sessionmanagement.t`	500	The length of time in milliseconds to be applied between autorecovery post requests to the server. This setting applies to all areas of the application where autorecovery is supported. The number must not be less than 1. Otherwise, it is treated as if throttling is disabled.

Procedure

The following steps outline how to set the configuration properties:

1. Log in to Social Program Management as a system administrator.
2. Click **System Configurations** and in the Shortcuts panel, click **Application Data** > **Property Administration**.

3. In the **Name** field, enter the property name, for example `Enable autorecovery` and click **Search**.
4. Select the **…** icon for the property.
5. Select **Edit Value** to update the value.
6. Click **Save**.
7. Click **Publish**.
8. Restart the server.

Configuring contact logs

Caseworkers use the contact log to record and manage interactions that are carried out during an investigation or as a follow up to an incident. You can configure system properties to control the functions of contacts that caseworkers record and maintain in the contact log.

> 8.0.3.0
>
> **Note:** After you enable the following properties, any existing contact narratives that were recorded within the configured edit period are editable only by their author until the editable period ends:
>
> - `curam.contactlog.narrative.edit.enabled`
> - `curam.contactlog.narrative.edit.period.length`
>
> When the editable period ends, all caseworkers can append to the narrative.

Table 2: Contact log configuration properties

Property	Display Name	Default value	Description
8.0.3.0 8.0.3.0 Enable Contact Log Subject	`curam.contactlog`	false	Defines whether the Subject field is enabled or disabled for contacts in the contact log. If you set the value to true, a Subject field is displayed on contact pages so that caseworkers can record a subject for contacts and search for contacts by subject.
8.0.3.0 Enable Editable Contact Narrative in Contact Logs	`curam.contactlog.`	false	Defines whether contact narrative editing is enabled or disabled. If you set the value to true, caseworkers can edit the narrative for contacts that they create or append to for a preconfigured time period. If you set the value to false, caseworkers cannot update the original narrative for a contact and can append a narrative update only.
8.0.3.0 Contact Narrative Editable Period Length	`curam.contactlog.`	72	Defines the length of time in hours during which caseworkers can edit the narrative for contacts that they create or append to in the contact log. The number of hours must be a whole number greater than 0. When the time period expires, the caseworker can no longer edit the narrative and can append a new narrative to the contact only.
curam.cases.maxnocases.contactlogs.`maxnocases.c`		100	Defines the maximum number of contacts that are displayed in the contact log.

Property	Display Name	Default value	Description
Display note history in descending order	curam.miscapp.notehistory...	YES	Defines whether records in the contact narrative history in the contact log and the note history are displayed with the most recent record shown first.

Procedure

The following steps outline how to set the configuration properties:

1. Log in to Social Program Management as a system administrator.
2. Click **System Configurations** and in the Shortcuts panel, click **Application Data** > **Property Administration**.
3. In the **Name** field, enter `Contact Log` and click **Search**.
4. Select the **…** icon for the property.
5. Select **Edit Value** to update the value.
6. Click **Save**.
7. Click **Publish**.

Configuring notes

Caseworkers can enter notes to store additional information that relates to a participant, case, incident, issue, or outcome plan. You can configure notes by setting application properties.

> **Note:** ▪8.0.3.0▪ After you enable the following properties, any existing notes that were created within the configured edit period are editable only by their author until the editable period ends:
>
> - `curam.misapp.editnoteenabled`
> - `curam.misapp.editnoteperiod.length`
>
> When the editable period ends, all caseworkers can append to the note.

Table 3: Notes configuration properties

Property name	Display name	Default value	Description
Client Merge - Merge Notes	curam.participant.clientmerge.mergenotes	YES	Defines whether notes merging is enabled or disabled when caseworkers merge client information from a duplicate record to the master record.
Display note history in descending order	curam.miscapp.notehistory...	YES	Defines whether records in the note history and the contact narrative history in the contact log are displayed with the most recent record shown first.
▪8.0.3.0▪ Enable Editable Note Text	curam.miscapp.edi... na...	false	Defines whether note text editing is enabled or disabled. If you set the value to true, caseworkers can edit notes that they create or append for a preconfigured time period. If you set the value to false, caseworkers cannot update the original note and can append a note update only.

Property name	Display name	Default value	Description
8.0.3.0 Notes Editable Period Length	curam.miscapp.edi...	72	Defines the length of time in hours during which caseworkers can edit notes that they create or append. The number of hours must be a whole number great than 0. When the time period ends, the caseworker can no longer edit the note and can append new note text only.
Length of truncated note text	curam.miscapp.truncatedno...	100	Defines the number of characters that note text must exceed before it truncates on the note list page.

Procedure

The following steps outline how to set the configuration properties:

1. Log in to Social Program Management as a system administrator.
2. Click **System Configurations** and in the Shortcuts panel, click **Application Data** > **Property Administration**.
3. In the **Name** field, enter Note and click **Search**.
4. Select the **…** icon for the property.
5. Select **Edit Value** to update the value.
6. Click **Save**.
7. Click **Publish**.

Configuring participant nicknames for search

Configure a thesaurus of nicknames for an individual.

The thesaurus allows the organization to define common nicknames that are associated with a name. For example, a person with the name "James" may also go by the name "Jimmy" or "Jamie". Defined nicknames can be used as search criteria when searching for a person and/or prospect person. The nickname search default setting is set in the administration application via property settings.

For more information on searching for persons by nickname, see the *Participant Guide*.

1.3 Configuring case audit selection queries

Case audits are used to examine and evaluate cases.

The random list of cases which are produced for a case audit are generated using selection queries. A selection query consists of an SQL statement and selection criteria used to validate the query and return information from the database. Two types of selection queries exist: fixed and dynamic.

A dynamic selection query is a flexible way to produce a list of cases for an audit. The coordinator can choose one or any combination of criteria to produce the list of cases. For example, the audit coordinator can select to generate a list of all cases with a status of open. Alternatively the audit coordinator could choose several criteria to produce the list of cases. For example, criteria of case start date and gender would return a more specific group of cases.

A fixed query is less flexible than a dynamic query, in that the values for the criteria form part of the query. An audit coordinator does not input the parameters for a fixed query. Fixed queries are reusable however, and are easier to run as no criteria selection by the audit coordinator is necessary. An example of a fixed query would be 'All open cases for males aged 18-35'.

For more information on selection queries, and case audits in general, see the *Case Audits Guide*.

For more information about selection queries and the SQL statements required to run a selection query, see the *Case Audits Developer's Guide*.

Creating and publishing a dynamic selection query

Dynamic selection queries can be created by a database administrator or a system administrator. Once created, they are associated with a case audit configuration by an administrator. Development effort is required to produce the new selection criteria page that an audit coordinator uses before the new selection query can be associated with a case audit configuration. A sample dynamic query is provided for each of the standard case types: integrated case, benefit product delivery, liability product delivery, and investigation case.

Page names for the manual search and random search pages are required when creating a dynamic selection query. These are the pages that the audit coordinator views when creating a list of cases for an audit. The system administrator must also enter the SQL statement for the selection query which is used to query the database for the list of cases to be returned.

As part of creating a selection query, selection criteria are recorded to ensure the query is valid. The system administrator then publishes the selection query, making it available to be added to a case audit configuration by an administrator. This allows an audit coordinator to generate a list of cases for audit using the selection query. The selection criteria are used to return the list of cases.

When configuring a case audit, an administrator must associate only one predefined dynamic query with a case audit configuration.

Creating and publishing a fixed selection query

Fixed queries are used in conjunction with dynamic queries. If configured, an audit coordinator can choose which type of query to use when generating the list of cases for an audit. Page names do not have to be specified when creating fixed queries because fixed queries are predefined. Audit coordinators are not required to enter any parameters for selection criteria that make up the query; therefore the pages to display the selection criteria are not required.

Otherwise, fixed queries are created similarly to dynamic queries, with an SQL statement that is validated using selection criteria. Once published, the administrator can then associate the fixed query with a case audit configuration. An audit coordinator can choose any fixed query that has been configured for a case audit. When run as part of the case list generation of an audit plan, this will return a list of cases for the audit coordinator in the runtime application.

An administrator can associate one or more fixed queries with a case audit configuration.

1.4 Configuring communication templates

Two types of communication templates are supported, Microsoft® Word and XSL templates.

For more information about communication templates, see the *Communications overview*.

Managing Microsoft® Word templates

Microsoft® Word templates are basic document templates. The templates can be edited so that templates are personalized for individual client communications. The template can be browsed for locally and uploaded. When you upload a Microsoft® Word template, you must complete the following tasks:

• Enter a name and template document ID for the template.
• Set a locale for the template.

When the caseworker creates a Microsoft® Word communication, the caseworker can select from various templates based on the locale of the applicable participant.

Fields are inserted into a Microsoft® Word template so that data such as the correspondent address information can be automatically populated in the Microsoft® Word communication when the communication is created.

> **Note:** Development work is required to insert fields with client data into a Microsoft® Word template. For more information, see Inserting fields into Microsoft® Word templates.

Managing XSL templates

XSL templates use a combination of XML and XSL stylesheets. XSL templates are used to generate pro forma documents and letters that are printed by the application. XSL stylesheets are used to format the XML data for printing. XSL templates can be created by using any XSL editor. XSL templates can then be uploaded and stored in the application database. When you upload an XSL template, you must complete the following tasks:

• Enter a description and template document ID for the template.
• Set a locale for the template.

When the caseworker creates a pro forma communication, the caseworker can select from various templates based on the locale of the applicable participant. You can create only one XSL template by using the same template ID and locale.

XSL templates can be checked out and downloaded. Checking out the template ensures that previous versions of the template are not lost. Templates can be checked out by more than one person at a time. System administrators can ignore other check-outs of a template. Template version control ensures that temples are not accidentally overwritten. An XSL stylesheet developer is responsible for the creating and maintaining the XSL templates. When a new version is ready to be uploaded, the system administrator can select to check in and upload the XML file.

For more information about XML and about generating documents from XML and XSL templates, see the *XML Infrastructure Guide*.

Assigning communication templates to case and participant types

XSL and Microsoft® Word templates can be assigned to a defined case or participant type because certain templates might only apply for specific types of participants or cases. For example, an appeal decision template is only applicable to participants who are engaged in an appeal. Administrators can apply templates to specific case and participant types based on a category. For example, the category case includes a number of cases types, such as income support and screening. When configured, the template is only available to the caseworker when the caseworker creates communications for the specified case type.

Fields are inserted into a Microsoft® Word template so that data such as the address information can be automatically populated in the Microsoft® Word communication when the communication is created. For more information about how to insert fields with client data, see Inserting fields into Microsoft® Word templates.

1.5 Configuring batch processes

Batch processes are programs that process many records according to set parameters. Due to the potentially large processing nature of batch jobs, they are often scheduled by organizations for off-peak times.

Adding a batch process to the application

Before the batch process can be used at run time, the related process operation in the application model is made available as a batch process during development by assigning a batch stereotype to the process. When the model is generated, an SQL executable for the batch process is created. This executable can then be added to the application by a system administrator. A batch executable can be associated with a single batch process only. The system administrator selects the batch process from the list of available batch processes. A name and description should be added, and the type specified. Batch processes can be of a reporting batch type or an archiving batch type. The batch process type relates to a coded description of the batch process, which is used to group similar batches processes.

For more information on creating a new batch process, see the *Developing Batch Processes Guide* guide.

Grouping batch processes

Batch process groups organize batch processes into logical groups. For example, financial batch processes can be grouped so that users do not have to search the entire list of batch processes to find a set of financial batch processes to run. Batch processes are grouped together simply by adding the batch processes to the same group and assigning a name for this batch process group. Batch process groups provide flexibility for an organization to manage and maintain their list of batch processes; batch processes can be grouped according to the needs of the organization.

Submitting a batch process for execution

Batch processes can be submitted for execution by selecting to run a batch process from the list of available batch processes. Depending upon the batch process, a number of parameters must be entered before the batch can be run. The batch request is then processed when the batch launcher is run. For more information about required parameters and running the batch launcher, see the *Developing Batch Processes Guide* guide.

You can define the values for set parameters when you submit a batch process. An example of a batch process is `DetermineProductDeliveryEligibility`. It is used to activate a large number of cases simultaneously and is therefore run as a batch process to defer this case processing to off-peak time. The batch process is configured to accept the parameter `product`. Setting the parameter to a specific product means that only cases of that product is processed. Note that the values for some parameters must be set in order for a batch process to run (setting other parameters is optional). A default value might also be set for a parameter, this apples every time the batch process runs unless you set a different value.

The order in which batch processes are submitted must also be considered as some batch processes do not function unless others are run previously. For example, `DetermineProductDeliveryEligibility` must be run before `GenerateInstructionLineItems` because instruction line items can only be generated for cases that are already activated.

After the system administrator has submitted the batch processes, it is held in a batch queue until the batch launcher runs. The batch launcher is a separate program that executes the batch processes in the order in which they were submitted. Note that batch jobs can have a processing date specified. Typically the system date is used as the business processing date. Where the processing date is specified, this date overrides the system date.

Creating a batch process error code

Batch process error codes specify the error codes that are returned by the application batch launcher if a batch process fails. Information that is recorded for a batch error code includes the batch error code ID and the batch error code. When a batch process fails, it outputs an error message, which is passed to the application batch launcher. The application batch launcher searches for a batch error code that matches the ID on the error message. If found, the application batch launcher then initiates the action that must be performed for the particular error. These actions are configured by an application developer.

For example, if the error code ID returned by a failed batch process is `CANNOT_CONNECT_TO_DATABASE`, then the batch launcher compares that to all batch process error codes that are stored in the system. If `CANNOT_CONNECT_TO_DATABASE` is found, then the batch launcher retrieves the batch error code that is associated with this batch error code ID, e.g., "11". The batch launcher passes this batch error code to a task scheduler. The task scheduler then examines its own configuration files to determine what to do in the event of receiving error code 11. For information about he batch launcher and other aspects of batch process administration, see the *Developing Batch Processes Guide* guide.

1.6 Configuring security

Application security ensures that only valid users can access the application. It also defines specifically what a user can view and change in the application when logged in.

Security administration is divided into two main categories: authentication and authorization. Authentication ensures that only valid users can access the application by requiring that the user provide a valid user name and password at login time. Authorization is used to secure functions in the application once a valid user has successfully logged in. Authorization defines a user's ability to perform actions and to access information within the application.

For more information about users, security roles, security groups, and the development implementation of security in the application, see the *Server Developer's Guide*.

Securing application elements

A security identifier represents a protected resource. Every secured element in the application is given a SID that is unique across the entire application. SIDs secure administrative functions, user

interface fields, organization units, locations, case audits, and programs that are offered by the organization.

The most common type of security identifier is the functional SID, also known as a function identifier or FID. FIDs are used to secure business processes. An example of a function identifier is the FID assigned to the register person business process. Another type of security identifier is the field SID. Field security is used to secure specific information that is displayed in a field on an application page or set of pages. An example of a field SID is the SID used to secure the participant bank account balance field.

Securing application functions

Server functions are secured by using FIDs. During application development, when a method is made publicly accessible; a unique security identifier is automatically generated for that function. In the deployed application, the methods that are contained in the model are generated as server functions. If security for a process method is disabled at design time in the model, a function identifier is still generated but is not available for used. When functions are generated, FIDs can be created and added to the security hierarchy by a systems administrator by searching for a function and associating it with a FID. Only functions that are not already associated with a FID are available for selection. Any changes made to a FID takes effect only when the changes are published.

Securing application fields

Field security governs the user's ability to view information in specific fields. Like functions, every field in the application can be secured by using SIDs that a user must have in their security profile to view or access that field. During application development, developers create SIDs for fields that require security. By default, security is not set on a field - the developer must decide that a specific field requires a SID. The SID is then added to the database. This SID must still be added to the security hierarchy by a system administrator who also ensures that the SID is added to the appropriate user profiles. Any changes made to a SID take effect when the changes are published.

Securing organization units, locations, and programs

SIDS can be created by a system administrator to secure access to organization units, locations, case audits, and programs, including products and service plans. For example, a system administrator can create a SID of type product that can then be used by an administrator to secure read access to products of a particular type. Likewise, a system administrator can create a SID of type organization unit that can then be used to control which users are able to maintain information about a particular organization unit. Any changes made to a SID take effect when the changes are published.

For more information about organization, location, and product security, see the *Organization Administration Guide*, the *Location Administration Guide*, and the *Integrated Case Management Guide*.

Grouping Related FIDs and SIDs together

A security group is the grouping of a set of related security identifiers. This level in the security hierarchy allows an administrator to group the large number of security identifiers into a smaller number of manageable groups. Any users who have a specific security group that is assigned to their security role have access to all the resources represented by the security identifiers

belonging to the security group. For example, users are authorized to register a person when their security role includes the security group that includes the register person security identifier.

User security profiles

User security profiles are defined by a hierarchy of SIDs. SIDs apply to both internal and external users and are used to secure administrative functions. SIDS also secure programs offered by the organization, including products and service plans.

The primary focus of the user security profile is to ensure that all users are authorized to access the information that they need to carry out their jobs in the organization while at the same time restricting those users from accessing secured information. The secondary focus of the user security profile is finding a way to best manage these profiles so that the work of the system administrator does not become a repetitive task.

Identifying security roles

The first step in creating user security profiles is to identify the organization's required roles. Organization can be large, with many users, it makes sense to create security profiles for users who share the same security access. It is also important to distinguish the different skill levels between users with similar profiles. While both a trainee user and an advanced user both work with cases, there would be limitations to certain business operations that the trainee user could perform. For example, a trainee user is unlikely to conduct case reviews and would not be responsible for case approvals. Therefore, it is not only the primary functions that define a user role, but also the different levels of users. By organizing SIDs into a hierarchical structure, similar business processes that are shared between various users can easily be distributed without having to manually declare all of the secured elements for each user profile.

Limiting user access to application elements

Security profiles apply to both internal and external users. By organizing SIDs into a hierarchical structure, similar business processes that are shared between various users can be distributed without having to manually declare all of the secured elements for each role. Each security role can be made up of any number of security groups, which in turn are made up of related security identifiers. Any changes made to a security role must be published before they can take effect.

Authorization evaluates a user's access to secured elements in the application based on his or her user security role. Every authorized user is assigned a security role; therefore, it is possible to authorize every user against any secured element of an application. External users are more restricted than internal users in what they can access.

Optimizing authorization by using the security cache

The security cache is an in-memory structure that holds the security information associated with user roles. Security information is held in this cache to optimize the performance of the authorization process.

The cache is refreshed when the application reboots; it can also be refreshed when a system administrator uses the cache refresh facility. The cache must be refreshed whenever any changes have been made to the user roles. This includes changes to security identifiers, security groups, and security roles. However, the addition of a new user, if there are no other associated security changes (e.g. to roles or groups), does not require a security cache refresh.

Online security process logging

To assist with troubleshooting and analysis, capture and write online process details to the server log file.

To enable online security process logging, set the application property `curam.online.data.based.security.logging` to `TRUE`. The default value of this property is `FALSE`, which prevents this information from being written.

Each log file message consists of the string `SECURITY_DIAG` followed by:

- `operationName` - The type of security check that has taken place.
- `loggedInUser` - The name of the logged in user.
- `concernRoleID` - The ID of the current concern role, if relevant.
- `caseID` - The ID of the current case, if relevant.
- `locationID` - The ID of the current location, if relevant.
- `securityCheckType` - The type of security check for a case that has taken place, if relevant. If applicable, a value of 1,2 or 3 can be outputted.

  - `1` - Maintenance Security Check for a case.
  - `2` - Approval Security Check for a case.
  - `3` - Read Security Check for a case.
- `locationAccessType` - The location access type, if relevant. If applicable, a value of LA1 and LA2 can be outputted.

  - `LA1` - Maintain.
  - `LA2` - Read.
- `caseSecurityCheckKey` - A key that is used to identify records that have been added to the cache. Can be used to add and retrieve records from the cache, if relevant.
- `cacheDetails` - A code depicting the current status of the cache.
- `securityPermDetails` - A code depicting the current security permissions.

> **Note:** If a particular field is not relevant, for example if there is no locationID set for the current Note sensitivity security check, a value of `N/A` will be inserted in the log message.

There are a number of different codes that can be returned for the `cacheDetails` and the `securityPermDetails` fields. Each code gives information on the current status of the cache and security permissions respectively.

The following are the codes that can be inserted into the `cacheDetails` field:

- `CD1` - cache on, cache hit, cache not updated.
- `CD2` - cache on, cache not hit, cache updated.
- `CD3` - cache on, cache hit, cache updated.
- `CD4` - cache on, cache not hit, cache not updated.
- `CD5` - cache disabled.
- `N/A` - not applicable.

The following are the codes that can be inserted into the securityPermDetails field:

- `SPD1` - correct security permissions, not restricted, not readOnly.
- `SPD2` - correct security permissions, not restricted, readOnly.
- `SPD3` - correct security permissions, restricted, not readOnly.

- `SPD4` - correct security permissions, restricted, readOnly.
- `SPD5` - incorrect security permissions, restricted, readOnly.
- `SPD6` - incorrect security permissions, restricted, not readOnly.
- `SPD7` - incorrect security permissions, not restricted, readOnly.
- `SPD8` - incorrect security permissions, not restricted, not readOnly.
- `SPD9` - correct security permissions, not applicable, not applicable.
- `SPD10` - incorrect security permissions, not applicable, not applicable.

Below are some sample messages from the server log:

- SECURITY_DIAG Concern Sensitivity and Location Security Check,superuser,101,N/A,N/A,N/A,LA2,101LA2superuser,CD2,SPD1
- SECURITY_DIAG Note Sensitivity Check,superuser,N/A,-7619527619557457920,N/A,N/A,N/A,N/A,CD5,SPD9
- SECURITY_DIAG Case Security and Location Security Check,superuser,101,-3007841601130070016,N/A,3,LA2,superuser101-30078416011300700163superuser,CD1,SPD5
- SECURITY_DIAG Case Creation Security Check,superuser,101,2501,N/A,N/A,N/A,N/A,CD4,SPD9

The above messages indicate the following events respectively:

- A concern sensitivity and location security check has occurred. The logged in user is superuser and the concern role ID is 101. There are no relevant caseID, locationID or securityCheckType. The locationAccessType is a read. The caseSecurityCheckKey is 101LA2superuser. The cache is enabled, but the result was not retrieved from the cache. The newly retrieved security result was then added to the cache. The user has the correct security permissions and the related case or participant is not restricted or read only.
- A Note sensitivity check has occurred. The logged in user is superuser. There is no relevant concern role ID. The note's ID is -7619527619557457920 and there is no relevant locationID, securityCheckType, locationAccessType and caseSecurityCheckKey. The cache is disabled and there was no cache hit and the cache was not updated. The user has the correct security permissions and it is not relevant whether the related case or participant is restricted or read only.
- A case security and location security check has occurred. The logged in user is superuser and the concern role ID is 101. The caseID is -3007841601130070016 and there is no relevant locationID. The securityCheckType is a read security check. The locationAccessType is a read and the caseSecurityCheckKey is superuser101-30078416011300700163superuser. The cache is enabled and the security result was retrieved from the cache. The cache was not updated. The user does not have the correct security permissions, and the related case or participant is restricted and read only.
- A case creation security check has occurred. The logged in user is superuser and the concern role ID is 101. The caseID is 2501 and there is no relevant locationID, securityCheckType, locationAccessType and caseSecurityCheckKey. The cache is enabled but there was no cache hit, and the cache was not updated. The user has the correct security permissions and it is not relevant whether the related case or participant is restricted or read only

1.7 Configuring Business Intelligence

Business Intelligence (BI) provides decision support information for case workers, supervisors, and senior managers in the organization. The information needed for each role is different and these are reflected in the BI tools that are available to each role.

BI consists of embedded analytics, and interactive dashboards and reports. Embedded analytics are used to represent data that has been pulled from the database and displayed to the user. Interactive dashboards are used to publish graphically intuitive displays of information, including dials, gauges, and traffic lights style graphs. These displays indicate the state of the performance metric compared with a goal or target value. Report functionality is used to create formatted and interactive reports with highly scalable distribution and scheduling capabilities.

The Business Intelligence and Reporting Tools (BIRT) Report Designer is an Eclipse plug-in which allows developers to create custom BI reports which can then be imported into the application. A wide range of graphs using the BIRT charting engine is supported, as well as data listing. BI content can be displayed in the application pages when required, a licensable BI dashboard is also available to be used. Once the reports have been created and are available on the system, they can be displayed in the runtime application using the BIRT Report Engine which renders the report design. It can produce output in a number of formats including HTML and PDF. The aggregated data of a BI report is displayed in such a way that the user can interact with it.

For more information on building and deploying BI reports, please consult the `Cúram BIRT Developers Guide`.

Configuring Business Intelligence reports

BI comes with a number of preconfigured sample reports. These sample reports show the reports infrastructure and how to use it to read and deploy information held in the database. Development work is required to make the reports available in the runtime application. Once development is complete, the reports should be copied to the BI content directory on the application server.

BIRT report options that can be configured via the system administration application include:

Table 4: BIRT Report Configuration Options

This table describes BIRT report options

Report Configuration Option	Description
Report name	This is the display name of the report. Reports can have a number of different configurations with different display names.
Report file name	The actual file path to the report design on the report server.
Report category	The category is used to classify the report for search purposes, for example, case, participant, etc.
Report servlet	This is the report servlet used to render the report.

Report Configuration Option	Description
Parameters	These refer to the frame that contains the chart rather than the chart itself. The width and height can be set, as well as whether scrolling is allowed for the report or not. A border can also be optionally displayed around the report.

Additional parameters can be added to BI reports. These can be from a set recognized by BIRT or business specific parameters that the report can be programmed to handle.

Configuring the Business Intelligence Report Viewer

A BIRT viewer is available for use to display BI reports in the runtime application. Options that can be configured for the viewer include:

Table 5: BIRT Report Viewer Configuration Options

This table describes BIRT viewer options

Report Viewer Configuration Option	Description
Viewer name	The name of the report viewer engine.
Servlet	The report viewer configuration servlet used to render the reports. Each report configuration can have its own servlet setting. The default option is 'run'.
Report name parameter	This specifies the report viewer type, The default is '__report'.
Context	This defines how the URL for the report is built.
Root	Specifies the report root configuration string. This can include the http specifier, server name, and port the report server is running on.

Additional default parameters can be added to the report viewer but these need to be from a set that BIRT recognizes in order to be used by the report viewer.

1.8 Configuring a target system

Use the configuration options available in the system administration application to configure target systems.

A customer might have several different system installations within their environment. The application supports several services that can communicate and interact with these other systems. When using one of these services, the system initiating the interaction is known as the source system, and the systems with which it is communicating are known as target systems. For example, a customer might set up two separate installations of the application for two distinct business operations. The customer might share evidence data between the two systems to improve their operating efficiency. In this scenario the two systems can be set up to communicate with one another and use the Evidence Broker to share evidence.

Creating a target system

For the services on a source system to communicate with target systems, the system administrator first needs to set up and configure the details for the target system on the source system. This is achieved using a dedicated target system configuration tab in the system administrator workspace.

Adding a service to a target system

You can associate multiple services with a target system. A URL must be defined for every service that is associated with a target system. The URL identifies and interacts with the service in the target system. The URL is generated by combining the root URL of the target system, consisting of the system hostname and port, and the extension URL for the associated service. For example, a URL `http://shell.example.com:9082/<servername>/services/EvidenceBroker` can be generated for the Evidence Broker service in a target system by combining the root URL (`http://shell.example.com:9082/`) of the target system and the extension URL(`<servername>/services/EvidenceBroker`) of the associated Evidence Broker service.

When adding a service to a target system, you can specify a user name and password. The password is encrypted during the creation of the target system service. The plain text password is never stored and Merative™ Social Program Management only ever compares the encrypted values for authentication. Subsequent editing of the service does not retrieve the password. If you edit the service, it displays an empty box for the **Invoking User Password** field. For more information, see the *Security Guide*.

Configuration Transport Manager (CTM) is another example where target systems are used. In CTM a target system is used to support the automatic transport of configuration data between source and target systems. When defining the target system for CTM, the Configuration Transport Manager service is used.

1.9 Content management interoperability services configuration

Understand the configuration options specific to the integration of the application with a content management system. This includes the application properties that enable integration and the configuration of metadata.

When the application is enabled for integration with a content management system, the documents associated with attachments and communications are stored in and retrieved from the content management system. Metadata information about the document, such as document type, can be stored with the attachment documents in the content management system.

For more information about how the application can be integrated with a content management system, see the *Content Management Interoperability Services Integration Guide*.

Enabling integration with a content management system

Enable integration of your application with a content management system by using a group of application properties located under the **Application - Content Management settings** category.

The following application properties control the level of integration with a content management system:

- curam.cms.enable specifies whether the storage location for certain files should be in the configured content management system instead of the application database. When the application property is enabled, use curam.cms.attachment.enable and curam.cms.proforma.enable to control what files should be stored in the content management system.
- *curam.cms.attachment.enable* specifies whether files categorized as attachments should be stored in the content management system. This includes attachments that are associated with recorded communications and Microsoft Word communications.
- *curam.cms.proforma.enable* specifies whether the files categorized as Pro Forma communications should be stored in the content management system. This includes all files associated with Pro Forma communications with the exception of those Pro Forma communications created as a result of batch processing.

Configuring metadata for attachments

When documents that are associated with attachments are created within the application and stored in the content management system, metadata information about the document can also be stored. This includes attachment documents associated with recorded communications and Microsoft Word communications.

The metadata information that can be stored about the document depends upon the context in which the attachment was created, for example, if an attachment is created in the context of a case, information about the case in which the attachment was created can be stored with the document; however, if the attachment is created in the context of a participant, no case information can be stored, but participant information may be stored instead.

If information about a document, such as the document receipt date, is subsequently modified in the application, the related metadata information may also be updated. Whether or not an update can be made to a particular metadata element will again depend upon the specific context that resulted in an update to the attachment.

The application can be configured to store a number of pre-defined metadata elements with the attachment document. By default, each metadata element is enabled, and can be individually disabled so that the information will not be stored along with the document in the content management system.

Enabling or disabling a metadata element will not affect any metadata that was previously stored in the content management system. Any changes to the OOTB metadata configuration will only come into effect when future attachments are created or when existing attachments are updated by either preventing metadata elements from being stored or allowing additional metadata elements to be stored, depending on the configuration setting.

Each metadata element has a display name and a description that are displayed to the administrator. Multiple display names and descriptions can be created for each metadata element to provide multi-language support and the existing descriptions and display names can be modified if required.

The following metadata elements are available for documents categorized as attachments, including attachments associated with recorded communications and Microsoft Word communications:

Table 6: Metadata Elements

This table shows the metadata elements available for attachments.

Metadata Element	Description
Case Reference	The case reference number
Participant Reference	The participant reference number
Participant First Name	The first name of a person or a prospect person
Participant Last Name	The last name of a person or a prospect person
Participant Date of Birth	The date of birth of a person or a prospect person
Document Type	The type of document
Document Type Code	The code for the document type
Document Receipt Date	The date on which the document was received
Communication Date	The communication date for attachments associated with recorded communications

An organization may also choose to implement additional metadata elements to meet their business requirements. For more information about how additional metadata elements are supported, see the *Content Management Interoperability Services Integration Guide*.

1.10 Inserting fields into a Microsoft® Word template

You can create a Microsoft® Word template. The template includes fields with variable data, such as a correspondent's address. You can also write the server code that is used to populate the template with the variable data for when the user creates a communication that is based on the template.

Creating a Microsoft Word template

To create the Microsoft Word template, use Microsoft Word. You must be familiar with the application.

About this task

To create the Microsoft Word template, you insert fields as placeholders into the template. When a user creates the communication, the placeholders are replaced with variable data that is specific to a correspondent.

> **Note:** When a user creates a Microsoft Word communication by using a template, the data that is specific to the correspondent becomes part of the communication itself. For example, where the Microsoft Word template includes variables for the correspondent's name and address, the correspondent's actual name and address, not the variables, are stored as the communication text.

To insert a field in a Microsoft Word template that is replaced with the variable data that the server returns, complete the following eight steps.

Procedure

1. Open a new Microsoft Word document locally, that is, independent of the application.

2. Create new custom document properties for the following five fields:

   - **AddressLine1**
   - **AddressLine2**
   - **AddressLine3**
   - **personName**
   - **userName**

   For more information about creating fields, see the *Sample Microsoft Word template content* related link.

3. In the template, click where you want to insert a field.

4. On the **Insert** tab for the text group, click **Quick Parts**.

5. Click **Field**.

6. From the **Categories** list, select the document information category.

7. In the **Field** names list, select **DocProperty**.

8. From the **Field Property** list, select the field that you created.

Results

When a user inserts the placeholders for the variable data as fields in the Microsoft Word template, the content of the body of the communication, which is the same for all communications that are generated by using this template, can be directly added to the document. The file is then saved as a standard Microsoft Word document. The user can browse for and upload the document in the same way that the user browses and uses any Microsoft Word document.

Related reference

Example of a Microsoft Word template.

Writing server code to populate communication data

When you create the Microsoft Word template and the template is available within the application, the user can select the template when the user creates a communication.

The user then enters all the other required details, for example, the correspondent name, the communication name, and other details, to create the communication.

When the user opens the communication, the server action that is called returns the variable data. The variable data is inserted into the fields in the Microsoft Word template, that is, the actual name and address of the correspondent is returned.

The server action returns the variable data as an object of data type Binary Large OBject (BLOB). The object consists of name-value pairs. In the pairs, the name is the fields in the Microsoft Word template and the value is the data to insert for the field when the document is created.

Sample Microsoft Word template content
Example of a Microsoft Word template.

In the proceeding example, the following are the custom document properties:

- **AddressLine1**
- **AddressLine2**

- **AddressLine3**
- **personName**
- **userName**

In the Microsoft Word template, these five custom document properties are replaced with the correspondent-specific data that is retrieved from the server. The custom document properties vary in each communication. However, the content in the body of the template is the same for all the communications that are created by using the Microsoft Word template.

```
{ DOCPROPERTY AddressLine1 }
{ DOCPROPERTY AddressLine2 }
{ DOCPROPERTY AddressLine3 }

 Dear { DOCPROPERTY personName }

  An example of Microsoft Word template.

  Thanks,
  { DOCPROPERTY userName }
```

Figure 1: Sample Microsoft Word template content

Sample code to return data to populate a Microsoft Word communication and the structure of the returned object

Sample code snippet that illustrates how to write the code to build the values into a BLOB object and return it to insert the values into the Microsoft Word document. Sample code snippet that illustrates the structure of the returned object that is built by the system as binary name-value pairs.

Sample code to return data to populate a Microsoft Word communication

> **Note:** Values are inserted in the form of name-value pairs that use the `org.jdom.Element.`

The <NAME> attribute in the name-value pair is the name of the **DocProperty** that is inserted in the template. The <VALUE> attribute is the correspondent-specific data that replaces the field in the Microsoft Word communication that is created.

```
   org.jdom.Element rootElement = new org.jdom.Element("ROOT");
   org.jdom.Element fieldsElement = new org.jdom.Element
("FIELDS");

   org.jdom.Element fieldElement = new org.jdom.Element
("FIELD");
   fieldElement.setAttribute ("NAME", "personName");
   fieldElement.setAttribute ("VALUE", "James Smith");
   fieldsElement.addContent (fieldElement);

   org.jdom.Element fieldElement1 = new org.jdom.Element
("FIELD");
   fieldElement1.setAttribute ("NAME", "AddressLine1");
   fieldElement1.setAttribute ("VALUE", "1074, Park Terrace");
   fieldsElement.addContent (fieldElement1);
```

```
      org.jdom.Element fieldElement2 = new org.jdom.Element
("FIELD");
      fieldElement2.setAttribute ("NAME", "AddressLine2");
      fieldElement2.setAttribute ("VALUE", "Fairfield, Midway");
      fieldsElement.addContent(fieldElement2);

      org.jdom.Element fieldElement3 = new org.jdom.Element
("FIELD");
      fieldElement3.setAttribute ("NAME", "AddressLine3");
      fieldElement3.setAttribute ("VALUE", "UTAH");
      fieldsElement.addContent (fieldElement3);

      org.jdom.Element fieldElement4 = new org.jdom.Element
("FIELD");
      fieldElement4.setAttribute ("NAME", "userName");
      fieldElement4.setAttribute ("VALUE", "Caseworker");
      fieldsElement.addContent (fieldElement4);

      rootElement.addContent (fieldsElement);

      return new curam.util.type.Blob (
      new org.jdom.output.XMLOutputter
      .outputString(rootElement).getBytes());
```

Figure 2: Sample code to return data to populate a Microsoft Word communication

Structure of the returned object from the code sample

The proceeding example illustrates the structure of the returned object that is built by the system as binary name-value pairs from the preceding code snippet.

```
<ROOT>
<FIELDS>
<FIELD NAME= "personName", VALUE="James Smith" />
<FIELD NAME= "AddressLine1", VALUE= "1074, Park Terrace"/>
<FIELD NAME= "AddressLine2", VALUE= "Fairfield, Midway" />
<FIELD NAME= "AddressLine3", VALUE="UTAH" />
<FIELD NAME= "userName", VALUE="Caseworker" />
</FIELDS>
</ROOT>
```

Figure 3: Structure of the returned object from a code sample

1.11 Configuring the FILE_EDIT widget with Google Chrome and Microsoft™ Edge

The Microsoft™ Word Integration Assistant is supported on Google Chrome version 104 and later, and Microsoft™ Edge version 104 and later based on Chromium.

The Microsoft™ Word integration functionality for Google Chrome and Microsoft™ Edge is based around the extension and native messaging features of those browsers. To use the native messaging solution, no extra server configuration is required, but a browser extension and a client desktop Java™ application, that is, the Word Integration Assistant, must be installed on all client

computers. The Word Integration Assistant is supported only on Microsoft™ Windows™ client machines. For more information, see the *Installing the Word Integration Assistant* related link.

The following list outlines the extra software installations that are required:

1. For the browser extension, download the SPM File Edit Native Messaging Bridge extension from the Chrome web store and install the extension for either Google Chrome or Microsoft™ Edge. Note, the SPM File Edit Native Messaging Bridge extension supersedes the Cúram File Edit Native Messaging Bridge extension. For more information, see the *SPM File Edit Native Messaging Bridge* related link. To use the FILE_EDIT widget, users must install the browser extension.

2. The Word Integration Assistant installer file `WordIntegrationAssistant.msi` is distributed with the Cúram Client Development Environment (CDEJ). The installer file is in the following location in the CDEJ: `CuramCDEJ\lib\curam\installers\`. To use the FILE_EDIT widget, users must install this application.

   The installer file also creates a new registry key. For a per user installation, the default behavior, the registry key is created in `Computer\HKEY_CURRENT_USER \Software\Google\Chrome\NativeMessagingHosts \curam.fileedit.chrome.nativebridge`. For a per machine installation, the registry key is created in `Computer\HKEY_LOCAL_MACHINE \Software\Google\Chrome\NativeMessagingHosts \curam.fileedit.chrome.nativebridge`.

   The value of the new key is the qualified path to a `.json` file, for example, `%WORD_BRIDGE_INSTALL_DIR%\hostmanifest.json`, where `%WORD_BRIDGE_INSTALL_DIR%` is the environment variable the installer creates that is set to the installation location of the host application. The following snippet outlines sample `.json` content:

   ```
   {
     "name": "curam.fileedit.chrome.nativebridge",
     "description": "Word Integration Bridge App",
     "path": "launchbridge.bat",
     "type": "stdio",
     "allowed_origins": [
       "chrome-extension://xxxxxxxxxxxxxxxxxxxx/"
     ]
   }
   ```

   The system uses `launchbridge.bat` to start the host application. For information about the structure of the `%WORD_BRIDGE_INSTALL_DIR%` directory, see the *Installing the Word Integration Assistant* related link.

> **Note:** For the Word Integration Assistant for Google Chrome and Microsoft™ Edge, the browser downloads the document to edit to the origin private file system that is beneath the browser's user data directory. The Word Integration Assistant does not support the customization of the user data directory for either browser type. Only the default user data directories for both browser types are supported.

Related tasks

The Word Integration Assistant is a client desktop Java™ application that communicates with the SPM File Edit Native Messaging Bridge browser extension, to enable users to edit Microsoft™

Word documents through the Social Program Management application. You must install the Word Integration Assistant on the client machines of all users who want to edit Microsoft™ Word documents by using the FILE_EDIT widget with either the Google Chrome browser or the Microsoft™ Edge based on Chromium browser. The Word Integration Assistant is only supported on Microsoft™ Windows™ client machines.

Related information

[SPM File Edit Native Messaging Bridge](#)

Installing the Word Integration Assistant

The Word Integration Assistant is a client desktop Java™ application that communicates with the SPM File Edit Native Messaging Bridge browser extension, to enable users to edit Microsoft™ Word documents through the Social Program Management application. You must install the Word Integration Assistant on the client machines of all users who want to edit Microsoft™ Word documents by using the FILE_EDIT widget with either the Google Chrome browser or the Microsoft™ Edge based on Chromium browser. The Word Integration Assistant is only supported on Microsoft™ Windows™ client machines.

Before you begin

> **Note:** The Word Integration Assistant client is not supported when a Roaming User Profile is enabled. For more information about the *RoamingProfileSupportEnabled* policy, see the Chrome and the Microsoft™ documentation.

Be aware of the prerequisite checks that the Word Integration Assistant installer performs. You can bypass the checks when you run the installation. However, we recommend that you allow the installer to run all the prerequisite checks so that it can identify any issues with the client machine configuration.

Some users might have deployment requirements where they want to bypass the prerequisite checks, and run only the installer to make the necessary system changes. For example, a customer might need to deploy the Word Integration Assistant before they deploy the other prerequisite software, such as Microsoft™ Word. You can run a script to perform the same system checks after installation, as described in *What to do next* after the procedure.

The following list outlines the prerequisite checks that the Word Integration Assistant installer performs:

- **Java™ Runtime Environment**
  The installer checks that Java™ Runtime Environment (JRE) version 1.8 or later is installed.
- **Microsoft™ Word**
  The installer checks that Microsoft™ Word version 14 or later is installed.
- **Browser checks**
  By default, the installer runs a series of browser checks for both Google Chrome and Microsoft™ Edge. You can bypass the checks, as described in the procedure.
  - **Google Chrome checks**
    - Checks that Google Chrome version 104 or later is installed.

- Checks that the Chrome native messaging policy permits the extension to communicate with the Word Integration Assistant. For more information, see the *Chrome Enterprise policy list* related link.
  - For per user installations, checks that the SPM File Edit Native Messaging Bridge extension is installed and enabled for one or more browser profiles.
- **Microsoft™ Edge checks**

  - Checks that Microsoft™ Edge version 104 or later is installed.
  - Checks that the Microsoft™ Edge native messaging policy permits the extension to communicate with the Word Integration Assistant. For more information, see the *Microsoft™ Edge Policies: Native Messaging* related link.
  - For per user installations, checks that the SPM File Edit Native Messaging bridge extension is installed and enabled for one or more browser profiles.

About this task

You can install the Word Integration Assistant by running either a full user interface installation, or a silent installation.

> **Note:** A full user interface installation displays an installation wizard and screen prompts that require you to move from one screen to the next. During a silent installation, the installer runs an installation without displaying a user interface. No prompts, messages, or dialog boxes are displayed. For more information about a silent installation, see the *Standard Installer Command-Line Options* related link.

The Word Integration Assistant installer has the following features:

- **Configurable installation location**

  For a silent installation, you can specify an installation location by using the `INSTALLDIR` property:

  - For a user installation, the `INSTALLDIR` property defaults to `%USERPROFILE%\IBM\WordBridge` if it is not configured.
  - For a machine installation, the `INSTALLDIR` property defaults to `%ProgramFiles(x86)%\IBM\WordBridge` if it is not configured.
- **Configurable JRE**

  You can specify that an existing JRE is used to start the Word Integration AssistantJava™ application.

  For a silent installation, use the `WORD_BRIDGE_JAVA_HOME` property to specify the path to an existing Java™ installation location. The installer uses the `WORD_BRIDGE_JAVA_HOME` property to create the WORD_BRIDGE_JAVA_HOME environment variable. If you do not configure the `WORD_BRIDGE_JAVA_HOME` property value, the installer queries the Windows™ registry to detect whether Java™ is installed on the local machine. If the installer detects that Java™ is installed, the installer sets the value of the `WORD_BRIDGE_JAVA_HOME` property to the `JavaHome` location that is recorded in the registry.
- **Support for installation per machine**

  The default behavior is per user installation of the Word Integration Assistant. However, you can also install Word Integration Assistant per machine, as described in the procedure.

- **Installation folder structure**

  The following list outlines the structure of the installed Word Integration Assistant folders:

  - *lib* **folder**
    The *lib* folder stores all the Java™ archive (JAR) files that the Word Integration Assistant uses.
  - *utils* **folder**
    The *utils* folder stores all utility batch and PowerShell scripts, except for the *launchbridge.bat* script.
  - *log* **folder**
    The *log* folder is the default location for the debug log files in a per user installation. The *log* folder is not created in a per machine installation. See the WORD_BRIDGE_DEBUG_FLAG and WORD_BRIDGE_LOGS_DIR entries in the environment variables list.
  - *jacob* **folder**
    The *jacob* folder stores the dynamic link library (DLL) files that the Word Integration Assistant uses.
  - *launchbridge.bat*
    The *launchbridge.bat* file is used to start the host application.
  - *hostmanifest.json*
    The *hostmanifest.json* file defines the native messaging host configuration. For more information, see the *Chrome Development Guide: Native Messaging* related link.

- **Environment variables**

  For a user installation, the installer creates the following environment variables at the user level. For a machine installation, the installer creates the following environment variables at the machine level, with the exception of WORD_BRIDGE_LOGS_DIR, as described in the WORD_BRIDGE_LOGS_DIR entry in the following list.

  - **WORD_BRIDGE_INSTALL_DIR**

    - Installation location for the Word Integration Assistant, where the value is an absolute file path.
  - **WORD_BRIDGE_JAVA_HOME**

    - Path to the JRE, where the value is an absolute file path.
  - **WORD_BRIDGE_DEBUG_FLAG**

    - Use to enable debug logging for the Word Integration Assistant.
    - The default value of WORD_BRIDGE_DEBUG_FLAG is -debug, which enables debug logging for Word editing sessions.
    - The output debug log file names have a format similar to *NativeMessagingBridge_%u_%g.log*, where *%u* and *%g* are numeric values, for example, *NativeMessagingBridge_1_1.log*. The log file name format prevents conflicts when you test the Word Integration Assistant concurrently with Google Chrome and Microsoft™ Edge.
  - **WORD_BRIDGE_LOGS_DIR**

    - For a per user installation, the installer creates the WORD_BRIDGE_LOGS_DIR environment variable at the user level. The default value is the *logs* folder in the installation location that is defined by the value of WORD_BRIDGE_INSTALL_DIR. You can change the value for a per user installation but it is not recommended.

- For a per machine installation, the installer does not create a WORD_BRIDGE_LOGS_DIR environment variable. By default, for a per machine installation, the log files are output to the location in the user's profile directory that is defined by the LOCALAPPDATA pre-existing Microsoft™ Windows™ environment variable. If you want to change the log files output location for users who share a machine level installation, you can do so manually after installation. See the *What to do next* section after the procedure.

- **WORD_BRIDGE_ENCODING_OPTS**

  - Stores the file and console encoding system properties that are passed to the Word Integration Assistant.
  - The default value is `-Dfile.encoding=Cp1252 -Dconsole.encoding=Cp1252`.
  - For more information about the encoding values, see the *-Dfile.encoding* related link.

- **Registry key**

  A successful installation creates a registry key, where the value of the key is the absolute file path to the *hostmanifest.json* file, for example, *C:\Users\user\IBM \WordBridge\hostmanifest.json*. The installation creates the registry key at the following location:

  - **Per user installation (default)**
    Computer\HKEY_CURRENT_USER\Software\Google\Chrome \NativeMessagingHosts\curam.fileedit.chrome.nativebridge

  - **Per machine installation**
    Computer\HKEY_LOCAL_MACHINE\Software\Google\Chrome \NativeMessagingHosts\curam.fileedit.chrome.nativebridge

Procedure

To install or to uninstall Word Integration Assistant, choose the appropriate option and enter commands in a command prompt:

- Run a full user interface installation:

  - To install per user, either double-click the MSI file to run the installer, or, alternatively, enter the following command:

    ```
    msiexec /i WordIntegrationAssistant.msi /l*v FullUIPerUserInstall.log
    ```

  - To install per machine, enter the following command as an administrator:

    ```
    msiexec /i WordIntegrationAssistant.msi ALLUSERS=1 /l*v
      FullUIPerMachineInstall.log
    ```

  For a full user interface installation, you can bypass prerequisite checks, although it is not recommended. For more information, see the Before you begin section.

  - You can bypass the check for a particular browser type by clearing the checkbox for that browser type when the wizard displays the supported browser types.
  - You can bypass all prerequisite checks only by running the previous `msiexec` commands and specifying the following MSI option in the command: `SKIP_PREREQ_CHECK=1`

- Run a silent installation:

  - To install per user, enter the following command:

    ```
    msiexec /i WordIntegrationAssistant.msi /qn /l*v SilentPerUserInstall.log
    ```

- To install per machine, enter the following command as an administrator:

```
msiexec /i WordIntegrationAssistant.msi ALLUSERS=1 /qn /l*v
  SilentPerMachineInstall.log
```

For a silent installation, you can specify the following MSI options in the command:

- To bypass Google Chrome checks, specify `CHECK_CHROME=0`.
- To bypass Microsoft™ Edge checks, specify `CHECK_EDGE=0`.
- To specify a different installation location, specify the following installation property value, where *PATH_TO_INSTALL_LOCATION* is the path to the installation location: `INSTALLDIR=PATH_TO_INSTALL_LOCATION`.
- To specify a different Word Bridge JRE, specify the following environment variable value, where *PATH_TO_JRE* is the path to the JRE: `WORD_BRIDGE_JAVA_HOME=PATH_TO_JRE`.
- To bypass all prerequisite checks, specify `SKIP_PREREQ_CHECK=1`

It is not recommended to bypass prerequisite checks. For more information, see the Before you begin section.

- To uninstall Word Integration Assistant, choose one of the following options:

  - In Microsoft™ Windows™ 10, remove the application from **Apps and Features**.
  - In earlier versions of Microsoft™ Windows™, remove the application by using **Add/ Remove Programs**.
  - For a full user interface uninstallation, enter the following command:

```
msiexec /x WordIntegrationAssistant.msi /l*v FullUIUnInstall.log
```

  - For a silent uninstallation, enter the following command:

```
msiexec /x WordIntegrationAssistant.msi /qn /l*v SilentUnInstall.log
```

What to do next

For the Word integration Assistant to operate correctly, you must ensure that the environment variables and the registry key that the `WordIntegrationAssistant.msi` installer creates are visible to the Google Chrome and Microsoft™ Edge processes that are subsequently launched. Therefore, after you run the `WordIntegrationAssistant.msi` installer, choose the appropriate option:

- After you run a per user installation, either close all Google Chrome and Microsoft™ Edge browser processes, or restart the machine.
- After you have run a per machine installation, restart the machine so that the system level environment variable and registry changes are detected by all operating system user accounts on the machine.

If you bypassed the prerequisite checks during installation, to run the same system checks after installation, run the following script: `%WORD_BRIDGE_INSTALL_DIR%\utils \mustGather.bat`. For more information and a detailed list of the scripts that the `mustGather.bat` script calls, see the next topic, *Troubleshooting the Social Program Management Word Integration Assistant*.

For a per user installation, to override the default log files output location after installation, configure the WORD_BRIDGE_LOGS_DIR environment variable value.

For users who are sharing a per machine installation, to change the log files output location do the following steps:

1. Create a log file folder with write permissions for the users.
2. Create a WORD_BRIDGE_LOGS_DIR environment variable and set the value to the path to the logs file folder that you created in the previous step.

If you want to use different log file folders for different users, then for each user you must create a WORD_BRIDGE_LOGS_DIR environment variable at the user level. Then, create the log file folder for the user at a location in the user's profile directory, for example:

- User A has a WORD_BRIDGE_LOGS_DIR user level environment variable whose value is `C:\Users\userA\WordIntegLogs`.
- User B has a WORD_BRIDGE_LOGS_DIR user level environment variable whose value is `C:\Users\userB\WordIntegLogs`.

Related information

[Standard Installer Command-Line Options](#)
[Chrome Enterprise Policy List: Native Messaging](#)
[Microsoft Edge Policies: Native Messaging](#)
[-Dfile.encoding](#)
[Chrome Development Guide: Native Messaging](#)

Troubleshooting scripts for the Word Integration Assistant

If any problems occur with the Word Integration Assistant after installation, you can run a set of troubleshooting batch scripts to examine the client machine configuration to identify any issues that might prevent the Word Integration Assistant from operating correctly. Also, if you bypassed prerequisite checks during installation of the Word Integration Assistant, you can use the `mustGather.bat` script to check your client configuration after installation.

The troubleshooting `mustGather.bat` script, which is in the `WORD_BRIDGE_INSTALL_DIR/utils` folder, calls the same batch scripts that the installer calls to run prerequisite checks. By default, the Word Integration Assistant installer runs a set of prerequisite checks to determine whether the client machine is correctly configured.

Prerequisite

Some of the scripts delegate to PowerShell scripts. Therefore, ensure that PowerShell is installed and enabled on the client machine.

Batch scripts

- **`mustGather.bat`**
  Calls the other batch scripts to run all prerequisite checks and prints a report to the standard output.
- **`checkWordInstalled.bat`**
  Calls `checkWordInstalled.ps1` to check that a supported version of Microsoft™ Word is installed.
- **`checkWordBridgeJavaHome.bat`**
  Checks that the WORD_BRIDGE_JAVA_HOME environment variable is set and points to a valid Java™ installation. The script also attempts to start Java™ to check the reported version.

- ***checkChromeInstalled.bat***
  Checks whether a supported version of Google Chrome is installed. The script uses PowerShell to retrieve the Google Chrome installation path from the registry and checks the version in the *chrome.exe* file.
- ***checkChromeExtensionEnabled.bat***
  Checks whether the SPM File Edit Native Messaging Bridge extension is installed and enabled for at least one Google Chrome profile. The script calls the *checkWordExtensionInstalledForChrome.ps1* PowerShell script.
- ***checkChromeNativeMessagingUserLevelHosts.bat***
  The script queries the registry directly to check whether the Google Chrome native messaging group policy permits user level hosts.
- ***checkChromeBlockList.bat***
  The script queries the registry directly to check whether the Google Chrome native messaging group policy permits the `curam.fileedit.chrome.nativebridge` native messaging host, that is, the Word Integration Assistant.
- ***checkEdgeInstalled.bat***
  Checks whether Microsoft™ Edge based on Chromium is installed. The script uses PowerShell to retrieve the Microsoft™ Edge installation path from the registry and then check the version of Microsoft™ Edge in the *msedge.exe* file.
- ***checkEdgeExtensionEnabled.bat***
  The script calls the *checkWordExtensionInstalledForEdge.ps1* PowerShell script to check whether the SPM File Edit Native Messaging Bridge extension is installed and enabled for at least one Microsoft™ Edge profile.
- ***checkEdgeNativeMessagingUserLevelHosts.bat***
  The script queries the registry directly to check whether the Microsoft™ Edge native messaging group policy permits user level hosts.
- ***checkEdgeBlockList.bat***
  The script queries the registry directly to check whether the Microsoft™ Edge native messaging group policy permits the `curam.fileedit.chrome.nativebridgenative` messaging host, which is the Word Integration Assistant.
- ***checkNativeMessagingBridgeRegistryKey.bat***
  The script checks for the registry key that is required by the browser extension to start the native messaging host. The MSI installer creates the key. The key's default value is the path to the *hostmanifest.json* file that permits the browser to start the native messaging host. The script also prints the contents of the *hostmanifest.json* file.
- ***stopWordBridgeNativeMessagingHost.bat***
  The script stops any hung Word Integration Assistant Java™ processes.

PowerShell scripts

- ***checkWordInstalled.ps1***
  The script uses PowerShell to query the Windows™ registry to verify that Microsoft™ Word is installed and is a supported version.
- ***checkWordExtensionInstalledForChrome.ps1***
  The script uses PowerShell to check whether the SPM File Edit Native Messaging Bridge extension is installed and enabled for at least one Google Chrome profile that belongs to the current user. The script checks the contents of the Google Chrome user data directory to find the profiles for which the extension installed and enabled. The script also calls the *curamHelperFunctions.ps1* PowerShell script.

- ***checkWordExtensionInstalledForEdge.ps1***
  The script uses PowerShell to check whether the SPM File Edit Native Messaging Bridge extension is installed and enabled for at least one Microsoft™ Edge profile that belongs to the current user. The script checks the contents of the Microsoft™ Edge user data directory to find the profiles for which the extension installed and enabled. The script also calls the *curamHelperFunctions.ps1* PowerShell script.

- ***curamHelperFunctions.ps1***
  The script runs helper functions that are used by the *checkWordExtensionInstalledForEdge.ps1* and *checkWordExtensionInstalledForChrome.ps1* PowerShell scripts.

Notices

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the Merative website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of Merative

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of Merative.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

Merative reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by Merative, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

MERATIVE MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Merative or its licensors may have patents or pending patent applications covering subject matter described in this document. The furnishing of this documentation does not grant you any license to these patents.

Information concerning non-Merative products was obtained from the suppliers of those products, their published announcements or other publicly available sources. Merative has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-Merative products. Questions on the capabilities of non-Merative products should be addressed to the suppliers of those products.

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

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

The licensed program described in this document and all licensed material available for it are provided by Merative under terms of the Merative Client Agreement.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to Merative, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. Merative, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. Merative shall not be liable for any damages arising out of your use of the sample programs.

Privacy policy

The Merative privacy policy is available at https://www.merative.com/privacy.

Trademarks

Merative ™ and the Merative ™ logo are trademarks of Merative US L.P. in the United States and other countries.

IBM®, the IBM® logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

Adobe™, the Adobe™ logo, PostScript™, and the PostScript™ logo are either registered trademarks or trademarks of Adobe™ Systems Incorporated in the United States, and/or other countries.

Oracle and Java are registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft™, Windows™, and the Windows™ logo are trademarks of Microsoft™ Corporation in the United States, other countries, or both.

UNIX™ is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.