Merative™ Social Program Management

Logging and monitoring

Clients are responsible for ensuring their own compliance with various laws and regulations, including the European Union General Data Protection Regulation. Clients are solely responsible for obtaining advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulations that may affect the clients' business and any actions the clients may need to take to comply with such laws and regulations. The products, services, and other capabilities described herein are not suitable for all client situations and may have restricted availability. Merative does not provide legal, accounting or auditing advice or represent or warrant that its services or products will ensure that clients are in compliance with any law or regulation.

This document is intended to provide guidance to help you in your preparations for GDPR readiness. It provides information about features of this offering, and aspects of the product's capabilities, that may help your organisation with GDPR requirements. This information is not an exhaustive list, due to the many ways that clients can choose and configure features, and the large variety of ways that the product can be used in itself and with third-party applications and systems.

The GDPR and the security of processing

Article 32 of the GDPR requires data controllers and processors to implement technical and organisational measures to ensure that personal data is processed securely. A level of data security must be applied that is appropriate for the level of risk that is presented by the processing of the personal data. Article 32(4) states that:

> *"The controller and processor shall take steps to ensure that any natural person acting under the authority of the controller or the processor who has access to personal data does not process them except on instructions from the controller, unless he or she is required to do so by Union or Member State law."*

Strategies to consider when reviewing these requirements include the following:

- **Data minimisation**: The GDPR states that the controller shall implement appropriate technical and organisational measures for ensuring that, by default, only personal data that is necessary for each specific purpose of the processing are processed. For details about data minimisation, see the SPM GDPR information page.
- **Authorisation**: Enforces restrictions to privileged user database access and application functions. For details about authorisation, see the SPM GDPR information page.

- **Auditing**: Can be used to monitor access to and use of personal data. Creates an audit trail that can assist forensics. For details, see the later section about Social Program Management and auditing .
- **Logging**: System logs can be used to help investigate or mitigate a security breach, but can also contain personal data that requires adequate protection under the GDPR. For details, see the later section about Social Program Management and logging.

Social Program Management (SPM) and auditing

SPM supports two auditing methods:

A. AuditMappings classes
B. Table-level auditing

AuditMappings classes

Audit fields contain extra information about the modification history of each record. You can add audit fields to database tables for auditing purposes. Audit fields are available on entity and struct classes, and are updated only by certain entity operations; read operations are not supported.

The developer must set the information in the audit fields, and must include the following information:

- *Creation time*
- *Modification time*
- *Program ID*
- *User ID*

Audit fields consist of all the attributes of a special class in the input meta-model called AuditMappings. You can automatically add a field that corresponds to each attribute of this special class to the database table and, also, to all the standard details structs for the entity.

For detailed information about audit fields, see the Knowledge Center topic at AuditMappings classes overview.

Table-level auditing

Table-level auditing can be enabled using the **Database table-level auditing** option in IBM Rational Software Architect Designer. Auditing is supported on all stereotyped entity operations except ns, nsmulti, batchinsert, and batchmodify. The information that is captured by table-level auditing is stored in the AuditTrail database table.

Enable table-level auditing by switching on the **Database table-level auditing** option for an operation. Table-level auditing captures information such as the date and time of the transaction, and the ID of the user who invoked the transaction. Before and after audit information about the record is only captured if optimistic locking is switched on.

The following information is captured by table-level auditing:

- **Date and time**: The date and time of the transaction.
- **User ID**: The ID of the user who invoked the transaction.
- **Table name**: The name of the modified table.
- **Program name**: The functional identifier (FID) of the invoked transaction.
- **Transaction type**: Indicates whether the transaction was *online*, *batch*, or *deferred* and so on.
- **Operation type**: Indicates whether the operation was create, read, update, or delete.
- **Key info**: The key that is provided to this operation. The key can identify one or many records.
- **Details of changed data**: Logged details of the changed data in an XML format. The exact format of the XML is viewable in the JavaDoc details for the class *curam.util.audit.AuditLogInterface* in the doc/api directory of the SDEJ. The JavaDoc includes the names of the fields that are referenced by the details struct, the field types, the new version of the field data and, if optimistic locking is enabled, the old version of the field data.

By default, the captured audit information is written to the AuditTrail database table. Developers can also supply their own auditing handler by specifying a class that implements the *curam.util.audit.AuditLogInterface* interface.

An audit handler can also be used to raise events in a particular scenario. Developers can write and register event handlers, which are classes that initiate an action when an event is raised, and, optionally, event filters, which use logic to determine whether to start the handler for a specified event.

For more information, see the Knowledge Center topic at Table-level auditing.

Social Program Management (SPM)and logging

Logging in SPM allows information to be logged regardless of whether the program is being run in online or batch mode. The final destination of the trace information is highly configurable. It can be a log file associated with the application server, a stand-alone log file, a console, or even a database. Developers need to be aware of the level of tracing configured, and that personal data can be output to the logs depending on the level of tracing.

To check the current trace level setting, call the `curam.util.resources.Trace.atLeast(Trace t)` method, where the parameter to the method can be one of the following options:

- `curam.util.resources.Trace.kTraceOff`
- `curam.util.resources.Trace.kTraceOn`
- `curam.util.resources.Trace.kTraceVerbose`
- `curam.util.resources.Trace.kTraceUltraVerbose`

To specify the trace level for the application, set the `curam.trace` property to one of the following values:

- trace_on (corresponds to O in the following table)
- trace_verbose (corresponds to V in the following table)
- trace_ultra_verbose (corresponds to U in the following table)

Each option causes significant information to be written to the log file, and can have a significant impact on the performance of the application. The following table shows how the value of the `curam.trace` property determines the trace level by causing corresponding trace properties to be enabled.

Trace property name	**Description**	**`curam.trace` property value**
`curam.trace.servercalls`	Trace server method invocations by remote clients. The information includes the name of the user who is requesting the invocation.	O
`curam.trace.methods`	Trace all business object method invocation.	V
`curam.trace.method_args`	Memory dump arguments, including their types, to business object method invocations.	U
`curam.trace.sql`	Trace SQL statements run by entity objects.	V
`curam.trace.sql_args`	Memory dump results of SQL select statements.	U
`curam.trace.rules`	Enables logging of rules	U
`curam.trace.smtp`	Trace the messages that are sent to the mail server.	

For more information, see the Knowledge Center Cúram Configuration Settings topic.

Personal data in logs

Depending on the configuration of the application server, data can be collected and stored in the following types of logs:

1. Access logs
2. Error logs
3. Security audit logs, if applicable
4. XMLServer logs

The logs can contain personal information such as IP addresses. The logs can also contain other personal data if your application is configured to output that personal data. Personal data that is captured in system logs may be subject to the consent requirements of the GDPR. Refer to the consent management document on the SPM GDPR information page for more details.

Further Information

The GDPR text
http://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN

Merative™ Social Program Management - GDPR Information in documentation.

Cúram Modeling Reference