xUML Runtime Architecture and Transaction Concept

The xUML Runtime consists of a micro kernel adorned with several add-ons.

- The **Micro Kernel** is responsible for loading the components and it is separated from each the **F rontend** and **Backend Adapters** by an **Abstraction Layer**. The abstraction layers guarantee that the service implementations are independent of the frontend and backend protocols.
- Frontend adapters are mapping incoming requests to the frontend abstraction layer. Several protocols and products are supported (third party / OEM adapters.) The xUML Runtime contains a comprehensive set of native frontend / backend adapters like SAP, SOAP, Java, SQL, HTTPS, FTP, LDAP, etc.
- An overall **Security** module manages authentication of HTTP requests with standard Apache modules, role based access control and serves as a security interceptor for pre/post-processing.
- A **Monitoring and Administration** module contains a server console for configuration and simple monitoring (log, service state). An error occurring, the monitoring module forwards this error to a monitoring service specified on the service.

On this Page:

Related Pages:

- setTransactionID()

Transaction Concept

In general, the xUML Runtime supports implicit and explicit commit of transactions.

- Fast-running ACID transactions are supported by the concept of xUML Runtime sessions.
- Long-running transactions are supported via BPMN 2 or persistent state concepts.
- As a part of other transaction concepts, SAP tRFC is supported.

Fast-running ACID Transactions

The xUML Runtime works with the concept of **sessions**. Sessions are equivalent to units of work, that can be committed or rolled back depending on the status at the end of the session.

The following action types are affected on commit / roll back:

Action Type	Example
database access	insertion or deletion of database records
persistent state handling	creation of a persistent state object, sending of a persistent state signal, sending of conversation signals
JMS activities	sending or receiving of JMS messages with acknowledge mode **transacted**
POP3 activities	deletion of mails from POP3 server

A session corresponds to one concrete instance of an execution of a service, namely of

- SOAP services,
- timer services
- schedulers
- HTTP services
- SAP RFC services
- JMS services
- Java services

Each session may contain multiple actions of the above described types. Per default, a session gets committed implicitly by the xUML Runtime when it is terminated without any exception. In this case, all involved transaction resources are committed. Additionally, explicit commits inside a session are allowed for all types of service adapters described above.
If an exception occurred during session execution and if this exception is not being caught, the session gets implicitly rolled back. The xUML Runtime will then rollback all not committed transactions.
However, for all other actions not mentioned in the table above, the session concept does not apply, e.g. file manipulations, external service calls, system adapter calls, and more.

Transaction Id

Each session is identified by a transaction id.

HTTP header **X-Transaction-ID** identifies the transaction the call belongs to. You can set the transaction id manually with setTransactionID(). If not set, the Runtime will generate one.
This header will be passed through the callstack to identify all service calls that belong to a transaction.

Long-running Transactions

For long-running business processes, the xUML Runtime supports parts of the BPMN 2 compensation semantics. If processes are defined via state machines, the UML concept of history states is supported. Concerning both concepts it is important to regard that in practice they should be integrated into an overall concept of error handling to make sense at all (e.g. annulation processes).

Other Transaction Concepts

Some backend systems have their own transaction concept, e.g. SAP tRFC. SAP tRFC has been implemented into the xUML Runtime by the use of a state machine. The correctness of this implementation has been verified by SAP during a certification process.