Session Manager Specification

OpenEdge Framework Technical Design Document

OpenEdge Application Architecture Specification (OEAA)

Version 1.0

Version	1.0		
Spec Team Members	Shelley B. Chase	schase@progress.com	Progress
	Marian Edu	marian.edu@acorn.ro	Acorn IT
	Rom Elwell*	rome@issol.com	Innovative Software Solutions
	Mike Fechner	mike.fechner@consultingwerk.de	Consultingwerk
	Ganesh Iyer	gaiyer@progress.com	Progress
	Peter Judge	pjudge@progress.com	Progress
	Christopher Longo	chlongo@progress.com	Progress
	Paul Moberg	paulmoberg@quickenloans.com	Quicken Loans
	Mark Opfer	mopfer@dmsi.com	DMSi
	Simon L. Prinsloo	simon@vidisolve.com	Vidisolve
	Robin Smith*	rosmith@progress.com	Progress
	* *Denotes contributing author*		

TABLE OF CONTENTS

1. Introduction

The Common Component Specification (CCS) project is designed to simplify the development of modern business applications by defining the architecture and components of a modern application development framework for the OpenEdge platform. The CCS project identifies a set of specifications for the common components needed in developing modern business applications.

This document describes the Session Manager as one of the required common components of the OpenEdge Application Architecture (OEAA).

Each request made to the business services will provide a sealed Client Principal object (C-P) or a known Session ID that identifies the client making the request. The Session Manager is responsible for asserting the identity of the client against the application runtime session and by doing so ensure that the client is valid for the application runtime. Once the client identity is authenticated, the Session Manager is responsible for providing a particular client session to service the request and provide access to the client context data.

A client session has context information associated with it such as user identity, what branch the user is logged into, date and numeric formats, time zone, etc. The Session Manager must provide access to this client context data by instantiating a Client Context object, which will be defined later in this specification. The identity of this Client Context object is set via public property on the Session Manager and is assigned either the C-P or Session ID as its value.

When establishing the session, after the C-P or Session ID of the client request has been validated, the Session Manger must then establish the context data on the first client request, or re-establish on subsequent requests, based on the client identity.

Once the request has been completed the Session Manager must “end” or “reset” the session. The Session Manager must reset the application server runtime session to a “safe” state so that the identity and context data of the client from the last request is not left asserted against the session and its databases. *See section 2.5.1.2 for further details.*

1.1 Purpose

The purpose of this document is to describe the required functionality of a Session Manager within the OpenEdge Application Architecture and define the API required by a CCS compliant Session Manager.

This document can be referenced by application developers who are using a CCS compliant framework to understand the expected behavior and functionality of a Session Manager.

This document should be used by framework developers and implementers of CCS compliant frameworks as a definition of the minimum requirements and expected behavior of a Session Manager.

1.2 Scope

The scope of this document is to define the required functionality of the Session Manager within the OpenEdge Application Architecture.

The Session Manager is responsible for establishing the ABL runtime environment within which the client request will be executed. The Session Manager will also

provide access to the client context data by creating and providing access to a Client Context object.

The Session Manager is also responsible for returning the ABL runtime environment to a safe state once the client request has completed.

The scope of this document will also cover the definition of the Client Context object as a sub component of the Session Manager and used to provide access to the client context data. Definition describing how to access the context data is left to the implementation.

1.3 Definitions, Acronyms and Abbreviations

Business Service – Services specific to the purpose of the application.

CCS – Common Component Specification. *(Chase, Elwell, & Smith, 2016)*

C-P – A client-principal is a handle-based object that functions as a security token in an ABL application.

OEAA – OpenEdge Application Architecture. *(Chase, Elwell, & Smith, 2016)*

OERA – OpenEdge Reference Architecture. *(Ormerod, 2006)*

Manager – Service specific to the Common Infrastructure. (Chase, Elwell, & Smith, 2016)

Service – Self-contained unit of functionality

SOA – Service oriented architecture

1.4 Contents Overview

Section 1 is the introduction and includes a description of the Session Manager.

Section 2 provides the overview of the Session Manager.

Section 3 describes the interfaces and classes used by the Session Manager.

2. Component Overview

2.1. Component Description

The Session Manager is used by the Service Interface or the Application Server's Activate and Deactivate procedures to establish the ABL runtime environment within which the client request will be executed. When the request has completed, the Session Manager is again used to "end" the ABL runtime environment and return it to a "safe" state ready for the next request.

The Session Manager is responsible for establishing the ABL runtime environment within which the client request will be executed.

This will include:

- Ensuring that the client's identity is asserted against the environment in order to establish the client's identity for the request.
- Establishing the client context by creating and providing access to a Client Context object.

The Session Manager is also responsible for returning the ABL runtime to a safe state once the client request has completed.

This will include:

- Ensuring that a neutral or "low access rights" security token is asserted against the environment in order to remove the previous client's identity from the ABL runtime environment.
- Persisting the Client Context object from the current request so that any changes made to it are available for subsequent requests.

2.2. Component Architecture

2.2.1. Ccs.Common.IManager

As a manager within the OEAA, the Session Manager will implement the IManager interface which inherits the IService interface, and therefore, will implement the *initialize ()* method.

The Session Manager's *initialize* () method can be used to retrieve any configurations it may require in addition to initializing itself.

2.2.2. Ccs.Common.ISessionManager

The CCS requires a 'pluggable' model that allows for components from different vendors to co-exist. As such, the Session Manager will implement a defined ISessionManager interface.

The ISessionManager interface will enforce a read-only property called *CurrentClientContext* as a reference to a Client Context object that will provide access to the current client's context data. This property will be defined as an interface to allow for framework configuration to define the implementation of this object.

The ISessionManager interface will enforce *establishRequestEnvironment* () and *endRequestEnvironment* () methods used by the Service Interface or the Application

Server's Activate and Deactivate procedures at the start and end of the client request.

The *establishRequestEnvironment* () method will have an input parameter for passing the handle of a C-P object or a known Session ID string. Whilst this handle is available in the *session:current-request-info* reference for a client request running on an Application Server, having it as a parameter will allow for platforms not running on an Application Server, such as testing and batch processing, to establish a runtime session by passing a sealed C-P.

2.2.3. Ccs.Common.IClientContext

The Client Context interface is the definition of the current Client Context object made available by the Session Manager.

The IClientContext interface will enforce an *initializeContext* () method that will be used by the Session Manager to pass in either the handle to the client's C-P object or a known Session ID string to initialize the Client Context object.

The interface will define a read-only *contextID* property that will represent a globally unique ID for the client session from which the current client request originates.

The interface will also define a read-only *clientPrincipal* property that can publish the handle to the C-P object representing the identity of the client.

The IClientContext interface will enforce a *saveContext* () method that can be used by the Session Manager to have the Client Context object save its context data to a persistent store so that it can be reinstated on subsequent requests.

It is expected that the Client Context object would behave similar to a property manager by providing *get* and *set* methods for storing context data. This is an implementation detail of the Client Context object and is not part of this specification. The application code can cast the *CurrentClientContext* object to the known type before accessing any context data.

2.2.4. SessionManager

The Session Manager object will be instantiated during the application framework bootstrap process. The *initialize* () method will be called during this phase and the Session Manager is expected to fetch any configuration data that it may require and initialize itself.

2.2.5. ClientContext

The Client Context object is created and managed by the Session Manager and represents the client's context data. The application code can get access to the Client Context data through this object which is available on the *CurrentClientContext* property on the Session Manager

2.3. Component Package Definition

The Session Manager interface and Client Context interface will be a subset of the Ccs.Common package.

2.4. Component Property Data and Organization

In terms of the naming conventions, storage organization and interaction between components, the requirements found in the document *CCS Specification: OpenEdge*

Application Architecture Version 1 also apply to the components defined in this specification.

2.5. Component Run-time Characteristics

2.5.1. Session Manager

The Session Manager object will be instantiated during the application framework bootstrap process by the Startup Manager. The *initialize* () method will be called during this phase.

The Session Manager is responsible for managing the server runtime environment within which a client request is executed. Some of those responsibilities could be delegated to other managers, such as a Security Manager, for example. Because the first version of the CCS does not define these potentially nominated managers as required components, the Session Manager may assume responsibility for those tasks.

2.5.1.1. *establishRequestEnvironment () Method*

The *establishRequestEnvironment* () method will be called by the Service Interface or the Application Servers registered Activate procedure at the start of a client request. Each client request must have either a sealed C-P object or a known Session ID associated with it.

When using the C-P, the C-P would typically be available on the *SESSION:current-request-info* attribute via the *GetClientPrincipal* method. The handle to the client request's C-P is passed to the *establishRequestEnvironment* () method. The *establishRequestEnvironment* () method will assert the client's C-P against the runtime environment by either delegating the task to a Security Manager or simply using the ABL runtime Security-Policy handle or the ABL *set-db-client* () function to validate and assert the C-P. The goal is to validate the client request and then assert the client's identity against the ABL runtime environment and connected databases. This will ensure that other OpenEdge components such as Auditing and Multi-Tenancy have the required C-P identity.

If the assertion of the C-P fails, the client request should be terminated and an appropriate error thrown.

When using a known Session ID, the *establishRequestEnvironment* () method should validate the value of the Session ID. If this fails, the client request should be terminated and an appropriate error thrown.

The *establishRequestEnvironment* () method must instantiate the Client Context object. The *establishRequestEnvironment* () method will then call the *initializeContext* () method on the Client Context object passing in the handle to the clients C-P or the known Session ID. The Client Context object is then responsible for initializing its context. This could be for a first time request or re-establishing its state for subsequent requests. Any error thrown by the Client Context object during initialization must terminate the request and an appropriate error thrown.

Once the Client Context object has been established, access to this object is made available to the runtime session by assigning an object reference to the read-only *CurrentClientContext* property of the Session Manager.

2.5.1.2. *endRequestEnvironment () Method*

The *endRequestEnvironment* () method will be called by the Service Interface or the Application Servers registered Deactivate procedure at the end of a client request once the request has been processed.

It is expected that the Service Interface or the activate/deactivate procedures implement appropriate error handling so that the *endRequestEnvironment* () method can be reliably called regardless of errors thrown during the execution of the client's request.

The *endRequestEnvironment* () method must "reset" the runtime session to a safe state by deleting the current Client Context object and clearing the *CurrentClientContext* property (i.e. setting its value to "?" (null) and then asserting a safe or low-access C-P token against the session and connected databases so that the identity of the client from the last request is not retained. It is not possible to assert a "blank" user against the database from the ABL so the Session Manager will need to have access to or be able to create a sealed C-P. The information required to perform this task would likely be fetched during the *initialize* () method call.

Before deleting the current Client Context object, the state of the Client Context must be saved away so that any changes will be reinstated for subsequent client requests. The *saveContext* () method on the Client Context object is used for this purpose.

2.5.2. Client Context Object

The Client Context object represents the client context data for the client making the request. The current Client Context object is instantiated and made available by the Session Manager.

The Session Manager must allow for the application to supply the implementation of the Ccs.Common.IClientContext interface

2.5.2.1. *initializeContext () Method*

The *initializeContext* () method will be used by the Session Manager to pass in the client's C-P object or a known Session ID. If the C-P is passed, then the Client Context object will use the C-P as the identity of the requesting client and assign the handle of the C-P to the read-only *clientPrincipal* property. If the Session ID is passed, then the Client Context object will use that Session ID to determine the identity of the requesting client.

The *initializeContext* () method must assign a globally unique identifier to the *contextID* property. The purpose of the *contextID* is to uniquely identify the client context data associated with the client's session.

The *initializeContext* () method must then establish or re-establish the client context data. If it is the first request, then the client context data may be initialized. If it is a subsequent request, then the context data should be retrieved from a context data store.

2.5.2.2. *saveContext () Method*

The *saveContext* () method may be used by the Session Manager to notify the Client Context object to save its context data to a persistent store so that the context data can be reinstated on subsequent request.

It should be noted that in a modern browser based UI application client requests can be asynchronous and care should be taken not to lose changes made to the Client Context due to browser support for asynchronous messaging.

2.6. Component Error Handling

Any application errors encountered by the Session Manager during the scope of the *establishRequestEnvironment* () and *endRequestEnvironment* () methods, should throw an error based on the ABL AppError and terminate the request.

Any errors caught by the Session Manager from subcomponents should be preserved and thrown on so that framework implementers can get meaningful feedback on what went wrong.

Any System Errors should be thrown and handled by the framework interface.

2.7. Dependencies and interactions with other OERA common components

The C-P passed to the *establishRequestEnvironment* () method needs to uniquely define the identity of the client's session making the request. It is expected that the *sessionID* will uniquely identify the C-P within the life time of the web servers running instance.

If the C-P is created by an ABL runtime, then it is expected that the *sessionID* will be assigned a GUID.

The Client Context Object may make use of a Context Data Manager to retrieve and persist the context data.

3. Component Interfaces and Classes

3.1. Ccs.Common.ISessionManager

```
USING CCS.Common.IManager.
USING CCS.Common.IClientContext.
USING CCS.Common.ServiceLifeCycleEnum.

INTERFACE CCS.Common.ISessionManager INHERITS IManager:

    DEFINE PUBLIC PROPERTY CurrentClientContext AS IClientContext NO-UNDO GET.

    METHOD PUBLIC VOID establishRequestEnvironment( INPUT phClientPrincipal AS HANDLE ).
    METHOD PUBLIC VOID establishRequestEnvironment( INPUT pcSessionID AS CHARACTER ).

    METHOD PUBLIC VOID endRequestEnvironment( ).

END INTERFACE.
```

3.1.1. Public Instance properties

Name	CurrentClientContext
Description	Holds the reference to the current ClientContext object.
Type	Ccs.Common.IClientContext
Getter	Public
Setter	Not public
Unknown	This property will be unknown unless there is a valid client request.

3.1.2. Public Instance methods

Name	initialize *Inherited from Ccs.Common.IService*
Description	Initialize the SessionManager
Return Type	Void
Parameters	None
Exceptions	Progress.Lang.SysError Progress.Lang.AppError

Name	establishRequestEnvironment
Description	Establish the server session runtime.

Return Type	Void		
Parameters	Input	Handle	The handle to the Client Principal object representing the client's session identity
Exceptions	Progress.Lang.SysError Progress.Lang.AppError		

Name	establishRequestEnvironment		
Description	Establish the server session runtime.		
Return Type	Void		
Parameters	Input	Character	A known string representing the client's session identity
Exceptions	Progress.Lang.SysError Progress.Lang.AppError		

Name	endRequestEnvironment
Description	"End" the server session runtime and return it to a "safe" state.
Return Type	Void
Parameters	None
Exceptions	Progress.Lang.SysError Progress.Lang.AppError

3.2. Ccs.Common.IClientContext

```
USING CCS.Common.IService.

INTERFACE CCS.Common.IClientContext:

  DEFINE PUBLIC PROPERTY contextID AS CHARACTER NO-UNDO GET.
  DEFINE PUBLIC PROPERTY clientPrincipal AS HANDLE NO-UNDO GET.

  METHOD PUBLIC VOID initializeConext( INPUT phClientPrincipal AS HANDLE ).
  METHOD PUBLIC VOID initializeConext( INPUT pcSessionID AS CHARACTER ).

  METHOD PUBLIC VOID saveContext( ).

END INTERFACE.
```

3.2.1. Public Instance properties

Name	contextID
Description	A unique ID for the Client Context object.
Type	Character
Getter	Public
Setter	Not public
Unknown	This property will be unknown or blank until the object's context has been initialized

Name	clientPrincipal
Description	The handle to the client's Client Principal object
Type	Handle
Getter	Public
Setter	Not public
Unknown	This property will be unknown until the object's context has been initialized and a C-P supplied

3.2.2. Public Instance methods

Name	initialize() *Inherited from Ccs.Common.IService*
Description	Initialize the Client Context object
Return Type	Void
Parameters	None
Exceptions	Progress.Lang.SysError Progress.Lang.AppError

Name	initializeContext()
Description	Initialize the context data for the Client Context object

Return Type	Void		
Parameters	Input	Handle	The handle to the Client Principal object for the client request
Exceptions	Progress.Lang.SysError Progress.Lang.AppError		

Name	initializeContext()		
Description	Initialize the context data for the Client Context object		
Return Type	Void		
Parameters	Input	Character	The known Session ID for the client request
Exceptions	Progress.Lang.SysError Progress.Lang.AppError		

Name	saveContext()
Description	Save the context data to a persistent store
Return Type	Void
Parameters	None
Exceptions	Progress.Lang.SysError Progress.Lang.AppError

4. Component Design and Implementation

This and the following section should provide sufficient information for a developer to produce the component. The detailed content will depend upon the approach to the design process that is to be used.

5. References

Chase, S. B., Elwell, R., & Smith, R. (2016). CCS Specification: OpenEdge Application Architecture Version 1. Progress Software.

Ormerod, M. (2006). Defining The Openedge® Reference Architecture - Common Infrastructure. Progress Software.

6. Document Control

Title: Session Manager Technical Design Document

Version: 1.0

7. Document History

Date	Version	Author	Change Details

Outstanding Issues

Provide details of any design issues that remain unresolved at the date of issue of this document. Explain options, pros and cons, and give an estimate of which option is most likely. Outline impact of each option on the rest of the design.