Common Component Specification Technical Design Document

Business Entity

Version 1.0

Version	1.0		
Spec Team Members	Freddy Boisseau*	flb@smsgroup.com	Structured Management Systems
	Mike Fechner*	mike.fechner@consultingwerk.de	Consultingwerk
	Dustin Grau	dugrau@progress.com	Progress Bravepoint
	Peter Judge*	pjudge@progress.com	Progress Software
	Chris Koster	chrisk@mip.co.za	MIP
	Robin Smith*	rosmith@progress.com	Progress Bravepoint
	Carl Verbiest	cverbiest@cce.be	CCE

** 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. When these components are built as part of a modernization framework, application developers can concentrate more on the business logic of the application rather than on infrastructure and integration.

The CCS Specification: Business Entity Version 1.0 builds on the Progress® OpenEdge® Reference Architecture (OERA) blueprint and defines interfaces and behaviour to use when developing Business Entity components with OpenEdge. The Business Entity is a central service component for data retrieval and data processing in business applications. The Business Entity is designed to be compliant with services oriented architectures (SOA) in general and especially the OpenEdge Application Architecture Specification Version 1.0.

Business Entities executed on the OpenEdge Application servers have become the central data access component for various Progress Software products such as the JSDO, Telerik's Kendo UI and Mobile products, Rollbase and DataDirect OpenAccess SDK for OpenEdge over the OpenEdge 11 releases. Independent software vendors and framework providers are using different tailor-made or standard frameworks already that are using Business Entity Components as described in the OpenEdge Reference Architecture.

The CCS Business Entity Specification describes the Business Entity component in a way that will allow to developers to use Business Entities from different CCS compliant frameworks that support the Business Entity specification.

These CCS specifications can be used in multiple ways. Some vendors will provide complete framework implementations, supporting all the CCS specifications while others might provide partial, focused frameworks only supporting some of the specifications, and others will implement single components for use with a CCS-compliant framework. For example, a security component might be provided by a specific vendor that specializes in security while another vendor focuses on UI metadata. For this reason, every component will be versioned independently of the CCS architectural version. As long as the components follow the architectural specification, components compliant to that architecture should work together nicely. For this reason, component specifications MUST identify the CCS OEAA architectural version or versions for which they are compatible.

1.1 Purpose

A Business Entity is defined in the OVERVIEW OF THE OPENEDGE REFERENCE ARCHITECTURE (Sadd, 2007) as the most basic business component, which manages a related set of data representing a meaningful business object such as an Order or a Customer. The Business Entity defines the data using a logical schema, typically defined as a ProDataSet with one or more ABL temp-tables, which represents the data in the way best suited to the needs of the business logic, and to the user interface and other Service Requesters, regardless of how the data happens to be stored in a database or other source. Together with its Service Interface(s), the Business Entity defines all the entry points that can be accessed by a particular requester type, and holds all the business logic for the data.

From this definition it should be clear that most applications have a component similar (of not identical) in design and intent to a Business Entity, although it may have a different name.

Business Entities have become widely implemented over the past decade as a central component of the OpenEdge Reference Application (OERA). Purpose of this document is to define the behaviour, interfaces and interactions with business entities as a standard component of the OpenEdge Application Architecture (OEAA) allowing both interaction and interchangeability of components from different vendors or developers.

The goal of the CCS project is to define a prescriptive architecture and standard set of specifications for the common components used in business applications by engaging the OpenEdge community and leveraging its expertise in building the best enterprise business applications. Each specification will include the API definitions as Object-oriented ABL (OOABL) interfaces, the expected behavior and other collateral to sufficiently define the component.

1.2 Scope

This document describes the Business Entity Component of the Business Services Layer as defined in the OpenEdge Reference Architecture. The specification defines the API, expected behaviour and other collateral needed to successfully use and implement a CCS compliant Business Entity.

The Business Entity is a standard component in which application developers can implement business logic for accessing (read operations) and modifying (insertion, update and deletion) a set of data (represented by a set of temp-tables in a ProDataset) as well as implement additional operations related to the domain of the Business Entity (operations that may for instance encapsulate specific read, data manipulation and update operations).

This specification does NOT define interfaces between the Business Entity component and a Data Access component. The OpenEdge Reference Architecture (OERA) does recommend the separation of the Business Logic and Data Access that these two types of components provide.

The CCSBE specification team members strongly recommend the use of Data Access components for implementing the physical data access required by a Business Entity as well. However, we have decided that we are not going to make the implementation of a Data Access component (and thus the data access functionality) outside of the Business Entity in a separate Data Access component mandatory.

We are not going to specify the interfaces between Data Access and Business Entity components as part of the Business Entity specification document. In the OpenEdge Reference Architecture (OERA) only the Business Entity is supposed to be interacting with the Data Access layer.

1.1 1.3 Definitions, Acronyms and Abbreviations

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

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

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

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

Service – Self-contained unit of functionality

SOA – Service oriented architecture

1.4 References to external documents

Num.	Title (Applicability & Reference)	Author	Date	Issue
1	CCS Specification: OpenEdge Application Architecture	Shelley B. Chase Rom Elwell	February 2016	
2	CCS Specification: Service Manager (work in process)	TBD	2016	
3	Defining The Openedge® Reference Architecture - Common Infrastructure	Mike Ormerod	August 2006	
4	Overview of the OpenEdge Reference Architecture	John Sadd	January 2007	V1.0

1.5 Contents Overview

Section 1 is the introduction and includes a description of the project, applicable and reference documents.

Section 2 provides the component’s overview.

Section 3 contains the component’s component descriptions.

Section 4 contains guidelines to implementers

Section 5 includes the component’s revision history, outstanding issues, and action items

2 Component Overview

The Business Entity provides CRUD (create, read, update and delete) functionality to the data structure (typically a ProDataset) that the Business Entity manages. The Business Entity MAY further provide high level business functions through additional custom methods.

It is mandatory that the Business Entity is implemented without knowledge of any specific client user interface. Business Entities can be used on the Application Server as well as in a fat client ABL session. It is mandatory that a Business Entities is developed in such a way that it can be made accessible through a suitable Service Interface component from any AppServer consumer as well as ABL GUI or GUI for .NET clients.

The Interface to the updateData() method requires a ProDataset with before image support (temp-tables defined with a BEFORE-TABLE) and activated change tracking. Consumers not capable of providing a ProDataset with before image data require the Service Interface component to transform the updataData() message to comply with this requirement. The Business Entity is not required to provide alternative interfaces to make changes to the data.

The data structure and capabilities of a Business Entity can be described through the data catalog (originally implemented for the JSDO and OpenEdge mobile).

The Business Entity relies on the Service Interface component to translate between an individual type of consumer and the interfaces of the Business Entity itself.

The CCBE spec ensures the Business Entity read operations (section 2.4) provide the full functionality required for responding to the JSDO’s JPF (ablFilter, orderBy, top and skip) request.

2.1 Component Description

The Business Entity is a Service in the sense of the Service Manager component. The Business Entity implements the IService Interface. The Service Manager acts as the factory for Business Entity components.

A Business Entity is implemented around a ProDataset definition. This ProDataset is considered the primary data structure of the Business Entity used in the interface to all read and update operations as described below.

The Business Entity provides three types of methods to its callers:

- Read operations
  Read operations return a ProDataset to the caller. The caller MUST provide information to the Business Entity that allows for data selection. The caller can provide information about which tables of the Business Entity ProDataset should be populated and data that is required to support paging (page size, page number (through the number of records to skip), starting record identifier, number of records to return).
- Update operations
  Update (create, update, delete) operations of the Business Entity expect the Business Entity ProDataset with changes as a parameter as well as an optional additional parameter object. A primary function of updates is data validation. Validation errors are expected to be reported to the caller using the error attributes of the ProDataset/ProDataset buffers. As the Prodataset does only support returning a single validation error message on a per record basis this document defines how multiple error messages and additional error properties MUST be encoded.
- Named operations
  A frameworks Business Entity implementation MUST support named operations. Named operations provide a suitable method of encapsulating business logic in a Business Entity allowing multiple consumers to reuse this functionality. Named operations MUST follow the following convention: These operations MUST be implemented as public methods in the Business Entity. Those methods SHOULD optionally be receiving a single ProDataset (not necessarily the Business Entity primary ProDataset) as an INPUT or INPUT-OUTPUT or OUTPUT parameter and optionally receive a request object. They return a response object as well as the ProDataset through when passed in as an INPUT-OUTPUT or OUTPUT parameter. As the ABL does not support delegates we can only require this signature of named operations through convention in this specification and not enforce it through a compiler verifiable definition.

All methods are expected to be called on their own through a Service Interface. The Business Entity is not expected to rely on a sequence of calls by a consumer that might be required to prepare the state of the Business Entity to be prepared for an actual call. Exception to this rule is the initialize() method as required by the IService interface (Service Manager specification) which will be called by the Service Manager during the start-up of the Business Entity..

Furthermore, the Business Entity is not expected to keep any session or client specific context between calls into the public methods of the Business Entity. All session context specific data that a Business Entity might need to access SHOULD be obtained from the Session Manager component during each request.

It is the responsibility of the Service Manager to define the life time of a Business Entity as a business service component. Business Entities are by definition stateless business components. The Business Entity MUST be implemented in a way that allows the usage of a Business Entity component by different clients in a sequence of AppServer requests in a stateless fashion to ensure the highest possible runtime performance and scalability avoiding the need to load every business entity for each request.

Business Entities are however allowed to maintain a cache of data that can improve the runtime performance of the Business Entity. The Business Entity is responsible for ensuring the life time and scope of any cached data, including if required managing to switch between different tenants over multiple AppServer requests (typically flushing any cache when a relevant context switch occurs).

2.2 Component Architecture

2.2.1 Ccs.Common.IService

Specification of this interface falls in the domain of the Service Manager specification.

2.2.2 Ccs.BusinessLogic.IBusinessEntity

The IBusinessEntity Interface extents the IService Interface as defined by the Service Manager specification.

Each operation is implemented as a single method that is supposed to be invoked by consumers through a Service Interface component. Most methods expose a ProDataset as the parameter. While the read and update operations MUST be using the same ProDataset structure the additional named operations MAY operate on a different (more specialized ProDataset structure).

The IBusinessEntity interface only defines operations (methods) and no properties. Business Entities are not expected to maintain any state between the invocation of their operations.

Business Entities are not expected to require parameters passed to their constructor. This cannot be enforced through an Interface definition but is a requirement of the Service Manager component.

2.2.3 getDataset() method

The Business Entity MUST provide a method which returns an empty instance (no data but schema) of its primary ProDataset.

This method is intended to be invoked by the data catalog to describe the data structure. It is also intended for (more dynamic) consumers that may require to receive a structure of the ProDataset as the primary message for interacting with the Business Entity before reading or updating data.

This method does not require any input parameters.

The method is not intended to be called BY-REFERENCE. The caller of the getDataset() method is responsible for cleaning up the ProDataset when no longer required.

2.2.4 getData() method

Most efficient data retrieval is a key functional requirement of any Business Entity. The getData() method will operate on a single request parameter object. This request parameter object MUST provide all relevant query arguments as well as arguments describing the amount of data expected by the caller and information about the set of data requested.

The IGetDataRequest parameter object contains an array of IGetDataTableRequest objects describing the request details for the requested temp-tables of the primary ProDataset of the Business Entity.

METHOD PUBLIC IGetDataResponse getData (IGetDataRequest, OUTPUT DATASET-HANDLE)

2.2.4.1 Filtering using an ABL query string

The Business Entity MUST support filtering using an ABL query string provided by the caller. This query string MUST be provided in a form like

CustNum = 42

CustNum = 42 AND OrderStatus = “Ordered”

It is mandatory that those Query Strings are expressed against the fields and tables of the Business Entities primary ProDataset. The Business Entity cannot expect knowledge about the fields and tables present in the actual physical storage (typically a database) from its consumer. When the Business Entity (or its Data Access object) perform a mapping between the ProDataset schema and the schema of the physical storage the Business Entity (or its Data Access object) is expected to map the provided Query String for execution as well.

The Query String is provided for each requested table as part of the IGetDataTableRequest interface. The Query String MAY be empty or unknown value indicating that no filter is required or filter is provided using query predicates or as a named query.

2.2.4.2 Filtering using an array of query predicates

As not every consumer of a Business Entity may be capable of providing a valid ABL query string to the Business Entity the Business Entity MUST provide the capability of filtering on an array of query predicates including the ability to provide nested groups of query predicates.

A query predicate consists of

- the Join criteria that defines which boolean operator should be used to join the predicate to its predecessor in the list. Valid values are defined by the Ccs.BusinessLogic.JoinEnum (None, Not, And, AndNot, Or, OrNot). The values of None or Not are only applicable to the first entry in a list.
- a field reference either in the form of “FieldName” or “TableName.FieldName”
- an Operator identified by the Ccs.BusinessLogic.QueryOperatorEnum Enumaration
- a Value represented by a holder class or a list of Values represented by an array holder class for the InRange or InList operators

The IGetDataTableRequest object provides the reference to an IQueryGroup instance. The IQueryGroup is an array of IQueryEntry instances. Each IQueryEntry instance is either a Query Group (with a Join criteria) or a Query Predicate as described above.

This structure allows for flexible and practically unlimited nesting of Query Predicates.

2.2.4.3 Filtering using a named query with parameters

For more complex queries it can be desired that the Business Entity itself (or its Data Access component) build the actual query criteria. This is particularly useful when query arguments might be a result of the session context or session state:

- TodaysOrders
- Yesterday's orders of Customer 42
- Invoices of month May 2016

To support those queries, the Business Entity will expect an INamedQuery object as part of the request parameter. The INamedQuery interface will consist of a Name property that returns the Named Queries identified to the Business Entity as well as an Array of Query Parameters (MAY have zero entries,i.o.W. EXTENT = ?). Each parameter to a Named Query consists of an INamedQueryParameter instance with a

- Name property to identify the Parameter (CustNum, Month, Year)
- Value property that contains a reference to an ICharacterHolder, IDecimalHolder, IDateHolder, … instance.

Named Querys MUST also support paging. The consumer is expected to provide IGetDataTableRequest instances as part of the IGetDataRequest object that provide the required paging properties.

2.2.4.4 Paging

A Business Entity MUST support paging (sometimes called batching) to optimize data retrieval to a consumer. Within this specification we distinguish between paging based on row numbers and paging based on a row identifier with a value meaningful to the Business Entity. A row number is provided as an integer value and the record identifier is provided as a character value, typically received from the Business Entity in a previous call.

The Business Entity MUST support returning the data in flexible page sizes. The number of records (NumRecords property of the table request parameter) can be provided by the consumer. When the consumer provides NumRecords as 0 the Business Entity is expected to return all (remaining) records to the caller. When the consumer provides NumRecords as ? the Business Entity is expected to use a reasonable default value for NumRecords (can be 0).

If a specific consumer is incapable of handling the unknown value (?) for NumRecords the Service Interface is responsible to provide a suitable alternative representation of the "default number of rows" NumRecords setting.

When the caller provides a value for the Skip property of the request parameter the Business Entity is expected to skip the number of records and starts returning the resulting records from Skip + 1. A consumer can start requesting the first page with a value of NumRecords = 100 and Skip = 0 and the second page of records with NumRecords = 100 and Skip = 100.

Alternatively, the client can use paging based on record identifiers. Record identifiers can consist of record key values or actual database rowid's (e.g. provided by a DATA-SOURCE RESTART-ROWID function). Generally, a consumer would not need to interpret this value in any way. Paging is achieved in the following way:

A consumer requests the first 100 records by providing NumRecords = 100 and the PagingContext = “” or ?. Using the response object, the Business Entity will also return the record identifier for the follow up call in the NextPagingContext property. In order to receive the next 100 records, the caller will call again into the Business Entity providing the value of the previous responses NextPagingContext as the value for the PagingContext property of the follow up request parameter. When a getData() request does return the unknown value for the NextPagingContext the Business Entity indicates the caller that there is no further data available.

When paging is requested based on record identifiers the Business Entity MUST support negative values for NumRecords to support batching in backwards direction. When a negative value for NumRecords is provided without a PagingContext value the Business Entity MUST return the very last complete set of resulting records (or fewer records when the query selection does not return enough rows to fill a complete batch).

When a negative value for NumRecords is provided together with a PagingContext the Business Entity is expected to return the complete set of records prior to the previously received set (or fewer record when the query selection does not return enough rows to fill another complete set).

The ability to specify the Tables of a request in combination with the NumRecords property allows a user interface to use an Order Business Entity to retrieve a list of all orders without all other details and a single Order Records with Order Line and Item information for a detail view.

2.2.4.5 Custom Parameters/Request Context

The IGetDataRequest parameter to the GetData methods provides CustomParameter object. This object is of any type and MAY be used to provide further (custom) instructions or information to the Business Entity.

2.2.4.6 Response Object

The response object of the getData() method provides context information for required for retrieving the next or previous set of data from the Business Entity.

The IGetDataResponse provides an array of IGetDataTableResponse instances that return the name of the temp-table that is described by the instance together with the

- NextPagingContext
- PreviousPagingContext

used for paging as described above

The IGetDataResponse object further provides a reference to an optional custom response object.

2.2.5 getResultCount() method

The getResultCount method is used by consumers (like the JSDO/Kendo UI) that navigate data in a paging fashion and require to receive the number of expected records before (or in parallel to) a getData request.

As the ABL and the OpenEdge database are not particularly strong in evaluating the total number of results of an arbitrary query the Business Entity MAY return a data guess or cached and potentially no longer accurate response to the consumer.

It's the joint responsibility of a framework provider that provides a tool set for implementing Business Entities and the developer implementing actual Business Entity components to ensure that the getResultCount() method returns a result in a way that does not stress the application performance in an unacceptable way.

Knowing the total number of records is typically only required for good user experience and then a guess (over 10,000 records which is provided in under a second) typically provides better user experience than an exact result that may require minutes to be calculated.

The count method will use the same parameter object as the getDataMethod and will return the number of matching records for every requested table.

PUBLIC IGetResultCountResponse getResultCount (IGetDataRequest) .

Details on the IGetDataRequest parameter are described in section 2.2.4 and 3.4.

The result of the method contains an Array of IGetTableResultCountResponse object instances providing the result count per requested table (in the same order as the requested tables) using the following information:

- Table Name
- Result Count
- Exact Result (to distinguish guessed or caches results)

The getResultCount method is expected to throw an error if counting the query result is not reasonable and would risk a serious negative impact on the system performance.

2.2.6 updateData() method

The Business Entity MUST provide a method which allows a consumer to send an instance of the primary ProDataset of the Business Entity with modified records to process and typically persist in a database or other type of storage system.

The Business Entity will process modified records only (ROW-STATE = ROW-DELETED, ROW-MODIFIED or ROW-CREATED) and ignore unmodified rows (ROW-STATE = ROW-UNMODIFIED).

This requires that code invoking the ProDataset updateData() method is capable of handling ProDatasets with before-image. It's the responsibility of the Service Interface to provide a ProDataset that complies with this requirement when routing requests of consumers that are not capable of providing a ProDataset with before-image information.

METHOD PUBLIC Progress.Lang.Object updateData (INPUT-OUTPUT DATASET < Primary ProDataset >) .

and

METHOD PUBLIC Progress.Lang.Object updateData (INPUT-OUTPUT DATASET < Primary ProDataset >, poRequest AS IUpdateDataRequest) .

Return value	An object that MAY be used to return additional data to the caller. SHOULD be serializable to allow sending to various consumers. Primitive response values MUST be wrapped in a holder object. Can be unknown value.
Primary ProDataset	The ProDataset to be used as parameter to this method. It's used for INPUT and OUTPUT. As the INPUT it's expecting a ProDataset with modified records (see above). On the OUTPUT those records MAY have the ERROR and ERROR-STRING attributes set. The Business Entity MAY also return more or different records to the caller. In a Business Entity with Customer and Salesrep updating the Customer's Salesrep field MAY cause the Business Entity to return the updated Customer and the new matching Salesrep to the caller. SHOULD be passed BY-REFERENCE
IUpdateDataRequest	An optional object with request data for the method. The IUpdateDataRequest object provides a property describing the suggested CommitScope and a custom parameter object that might be used by the Business Logic to control custom behaviour.

The IUpdateDataRequest interface contains a CommitScope property. The property returns a value of the CommitScopeEnum and allows the consumer to suggest or a hint to the business entity about the transaction scope while processing multiple modified records. Supported values are:

All	All records in the Dataset are processed in a single database transaction
Row (Default)	One database transaction per table row

Table	One database transaction for all records in a single table
Nested	One database transaction per parent row and its child table and all grand-child records.

The Business Entity implementation is allowed to override and ignore this setting as the Business Entity and its potentially used Data Access object are solely responsible for the transaction scope.

The IUpdateDataRequest object further contains a custom request object which is intended to be used for additional business logic implementation specific request details.

Validation messages are returned using the ERROR and ERROR-STRING attributes of the ProDataset member records. When at least one record is marked with ERROR, the ProDataset MUST be marked with ERROR as well.

See further information on the section 2.5 about Component Error Handling in this document.

The ProDataset can be passed BY-REFERENCE from the caller to the Business Entity.

2.2.7 Named operations

The Business Entity MAY contain other custom public methods.

These methods encapsulate business logic beyond reading and updating of values (CRUD operations). While shipping an order might just require to update the OrderStatus and ShippingDate fields of the Order record - it is considered as bad practice to implement the shipment of an order in that way as this would require too much knowledge about shipping an order on the side of the consumer. Encapsulating this in a custom method will also allow to reuse exact the same logic for shipping an order from any consumer – including a unit test environment.

Examples for custom methods of a Business Entity:

- shipOrder
- cancelOrder
- customerCreditCheck
- bulkProcessOrders
- getInitialValues

Custom public method matching the requirements described in this section are called “named operations”. The pattern of the named operations described here will simplify exposing these methods through a generic service interface and describing them in the service catalog.

Named operations SHOULD optionally receive a ProDataset parameter (not necessarily the Primary ProDataset of the Business Entity) as an INPUT, INPUT-OUTPUT or OUTPUT parameter. Additionally, named methods MAY receive a single request object as an input parameter.

The signature for these named operations MUST be:

METHOD PUBLIC <ResponseObjectType> <NamedOperationName> ({INPUT|OUTPUT|INPUT-OUTPUT DATASET < ProDataset>} {, poRequest AS <RequestObjectType>}) .

ResponseObjectType	An optional object with response data of the method. SHOULD be serializable to allow sending to various consumers. Primitive response values MUST be wrapped in a holder object. Can be unknown value.
NamedOperationName	The actual name of the method, typically consisting of a verb and a noun, SHOULD be descriptive
Primary ProDataset	The optional ProDataset parameter to be used with this method. It’s up to the implementer of the custom method to define the ProDataset parameter either as INPUT or OUTPUT or INPUT-OUTPUT. Can be passed BY-REFERENCE
RequestObjectType	An optional object with request data for the method. SHOULD be serializable to allow receiving from various consumers. Primitive request values MUST be wrapped in a holder object.

A Business Entity that supports named operations MUST implement the ISupportNamedOperations Interface. This interface requires the getNamedOperations() method which returns an Array with the names of the invokable named operations.

2.3 Component Package Definition

The Business Entity Interface and the supporting interfaces (for request and response objects) are defined in the Ccs.BusinessLogic interface.

2.4 Component Property Data and Organization

The Business Entity is designed around its primary ProDataset. This primary ProDataset is returned by the getData() and getDataset() methods and expected as the parameter for the updateData() methods.

The data catalog originally implemented to support JSDO based clients describes this ProDataset.

Named operations MAY use different ProDatasets in their interface.

2.5 Component Error Handling

In any situation where a Business Entity method is not able to perform it's task it is expected to throw errors to its caller. The error message MUST state clearly the nature of the error preventing the Business Entity to perform the task and the error is expected to provide useful additional properties for analysis and logging. It is strongly advised to throw error objects with error messages and not with the legacy ReturnValue (e.g. using the AppError constructor with a CHARACTER and an INTEGER parameter instead of just a single CHARACTER parameter).

In the current revision the CCSBE specification will not enforce different error classes or interfaces to be used for different error scenarios.

Data Validation during update operations is considered an application function and a key behaviour of the (Updatable) Business Entity. As such validation errors (the consumer passes data for update to the Business Entity that does not match the requirements of the Business Logic such as empty fields, data values out of range, foreign key violations etc.) are not expected to be thrown to the caller. Validation error are not runtime errors; they are part of the business logic functionality of a Business Entity. These validation errors are returned to the caller using the mechanisms of ProDataset member records (the ERROR and ERROR-STRING attribute of the member record as well as the ERROR attribute of the ProDataset itself).

Validation messages returned through the ERROR-STRING attribute of the temp-table records are recommended to be encoded in such a way that the Message Manager (defined by CCS later) is able to return a localized error message to the consumer.

Business Entity implementation that should remain usable outside of a full blown CCS implementation SHOULD be prepared to return validation messages directly in a human understandable form as the Message Manager of the CCS may not be available.

As multiple separate validation messages (for different fields) MAY need to be returned those messages are expected to be returned as a JSON array. Each object in the JSON array MUST provide the following properties if applicable:

FieldName	The name of the field that caused this error. Leave blank if the error is caused by the whole record.
MessageStrings	The JSON string array of full unencoded error messages, one string per line.
MessageId	The numeric identifier of an error message
MessageGroup	The error message group
SubstitutionValues	The JSON string array of values to substitute placeholder (&1, &2, &3, …) in the message template identified by MessageId and MessageGroup

Severity	A string with is either “Info”, “Warning” or “Error”

The Business Entity SHOULD either set the MessageId and MessageGroup or the MessageStrings property in the message structure. MessageId and MessageGroup are typically passed on the client or the service interface to a Message Manager component for retrieving a localized message.

If one or more JSON Array elements for MessageStrings is provided the MessageId, MessageGroup and SubstitutionValues MAY be left empty.

The MessageId (message number) and MessageGroup are used to retrieve a message template string from a Message Manager.

Empty JSON properties are not supposed to be contained in the ERROR-STRING property of the ProDataset member record.

2.6 Dependencies and interactions with other OERA common standards

The Business Entity is a Service as defined by the Service Manager spec.

3 Component Interfaces and Classes

All parameter and return types that are documented with no package name are members of the Ccs.BusinessLogic package.

3.1 Ccs.BusinessLogic.IBusinessEntity

Inherits Ccs.Common.IService.

This is the main interface for a Business Entity. Is describes the mandatory methods for a read-only Business Entity. The interface inherits from the IService Interface (see specification of the Service Manager) as every Business Entity MUST be manageable by the Service Manager component.

```
USING Ccs.Common.*        FROM PROPATH .
USING Ccs.BusinessLogic.* FROM PROPATH .
USING Progress.Lang.*     FROM PROPATH .

INTERFACE Ccs.BusinessLogic.IBusinessEntity
    INHERITS IService:

    /*------------------------------------------------------------------------------
        Purpose: Returns an empty instance of the primary ProDataset to the caller
        Notes:   Used for catalog generation or initialization of dynamic user
                 interfaces etc.
        @param phDataset OUTPUT Dataset (not intended to be called BY-REFERENCE)
    ----------------------------------------------------------------------------*/
    METHOD PUBLIC VOID getDataset (OUTPUT DATASET-HANDLE phDataset).

    /*------------------------------------------------------------------------------
        Purpose: Performs a read request
        Notes:
        @param poRequest The IGetDataRequest instance with the getData request
parameters
        @param phDataset OUTPUT Dataset
        @return The IGetDataResponse instance
    ----------------------------------------------------------------------------*/
    METHOD PUBLIC IGetDataResponse getData (poRequest AS IGetDataRequest,
                                            OUTPUT DATASET-HANDLE phDataset).

    /*------------------------------------------------------------------------------
        Purpose: Returns the count of the total number of result records or a
                 Guess of the result count to the caller
        Notes:
        @param poRequest The IGetDataRequest instance with the getResultCount
                         request parameters
        @return The IGetResultCountResponse instance
    ----------------------------------------------------------------------------*/
    METHOD PUBLIC IGetResultCountResponse getResultCount (poRequest AS
IGetDataRequest).

END INTERFACE.
```

3.1.1 Public instance methods

Name	getDataset()

Description	Returns an empty instance of the primary ProDataset to the caller. This method can be invoked for data catalog generation or the initialization of dynamic user interfaces etc.. The caller of this method is responsible for disposing the Dataset from memory when no longer needed.		
Return Type	Void		
Parameters	OUTPUT	DATASET-HANDLE	An Empty instance with a copy of the Business Entity primary ProDataset
Exceptions	Progress.Lang.SysError Progress.Lang.AppError		

Name	getData()		
Description	Performs a read request and returns a ProDataset with the resulting data		
Return Type	IGetDataResponse		
Parameters	INPUT	IGetDataRequest	The request instance that contains all parameters to the getData call. The IGetDataRequest instance does also provide a custom parameter object instance. The definition of this custom parameter object is up to the implementer of the Business Entity.
	OUTPUT	DATASET-HANDLE	The Business Entity primary ProDataset with the data matching the IGetDataRequest. Can be called BY-REFERENCE
Exceptions	Progress.Lang.SysError Progress.Lang.AppError		

Name	getResultCount()		
Description	Returns the count of the total number of result records or a guess of the result count to the caller. The implementation MUST provide a well performing implementation.		
Return Type	IGetResultCountResponse		
Parameters	INPUT	IGetDataRequest	The request instance that contains all parameters to the

			getData call. The IGetDataRequest instance does also provide a custom parameter object instance. The definition of this custom parameter object is up to the implementer of the Business Entity.
Exceptions	Progress.Lang.SysError Progress.Lang.AppError		

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

3.2 Ccs.BusinessLogic.IUpdatableBusinessEntity

Inherits Ccs.BusinessLogic.IBusinessEntity

Interface for a Business Entity that provides update capabilities (create/delete/modify) to its consumers.

```
INTERFACE Ccs.BusinessLogic.IUpdatableBusinessEntity
    INHERITS IBusinessEntity:

    /*------------------------------------------------------------------------------
        Purpose: Stores data modifications in the persistent storage (typically a
                 database)
        Notes:   The output dataset will contain validation error messages in the
                 ERROR-STRING attributes of the record buffers. Records with
                 Errors will also have the ERROR attribute set to TRUE. When at
                 least a single record has a validation error, the ERROR attribute
                 of the ProDataset is assigned to TRUE as well
        @param phDataset INPUT-OUTPUT Dataset containing modified records to be
                         processed (should be passed BY-REFERENCE)
        @param poUpdateDataRequest The optional request object that allows to
                                   provide custom instructions to the method
        @return An optional response object returned by the method
    ----------------------------------------------------------------------------*/
    METHOD PUBLIC Progress.Lang.Object updateData
                                  (INPUT-OUTPUT DATASET-HANDLE phDataset,
                                   poUpdateDataRequest AS IUpdateDataRequest).

END INTERFACE.
```

3.2.1 Public instance methods

Name	updataData()		
Description	Stores Data Modifications in the persistent storage (typically a database). The consumer can provide a hint for the transaction scope to be used by the Business Entity (see the IUpdataDataRequest interface below). The Business Entity is in full control of the transaction scope and MAY ignore the hint provided by the consumer completely. The Business Entity MAY return an arbitrary response object from the updateData() method. This definition of this response object is up to the implementer of a specific Business Entity		
Return Type	Progress.Lang.Object	An arbitrary response object	
Parameters	INPUT-OUTPUT	DATASET-HANDLE	The ProDataset instance with modifications, MAY be passed BY-

			REFERENCE. The Business Entity will update the ERROR and ERROR-STRING attributes of the ProDataset and the affected ProDataset temp-table buffer (record) in case of validation errors. See section 2.5for details.
	INPUT	IUpdateDataRequest	The request object that provides the proposed commit scope and an arbitrary request object. The purpose of the arbitrary request object is up to the implementer of the actual Business Entity.
Exceptions	Progress.Lang.SysError Progress.Lang.AppError		

3.3 Ccs.BusinessLogic.ISupportNamedOperations

Interface for Business Entities that support Named Operations that are exposed to consumers.

Named operations are PUBLIC instance method of the Business Entity with custom functionality (e.g. ShipOrder, ValidateCustomer). The purpose of including the named operations in the Business Entity specification document is to provide a set of recommended signatures for named operations.

The Interface can be implemented by IBusinessEntity and IUpdatableBusinessEntity instances. As the Interface does not inherit from the IBusinessEntity interface, the interface can be used by other Business Logic objects (e.g. Business Tasks as well).

```
INTERFACE Ccs.BusinessLogic.ISupportNamedOperations:

    /*------------------------------------------------------------------------------
        Purpose: Returns the names of the supported named operations
        Notes:   Used for catalog generation and an entry point into reflection
        @return The array with the names of the invokable named operations
    ------------------------------------------------------------------------------*/
    METHOD PUBLIC CHARACTER EXTENT getNamedOperations ().

END INTERFACE.
```

3.3.1 Public instance methods

Name	getNamedOperations()	
Description	Returns the names of the suppported named operations. Used for catalog generation and an entry point into reflection.	
Return Type	CHARACTER EXTENT	The array with the names of the Named operations
Exceptions	Progress.Lang.SysError Progress.Lang.AppError	

3.3.2 Named Operations

Named Operations are not documented as part of the Interfaces section in this document as the ABL lacks the capabilities of describing those methods through delegate types. See section 2.2.7 for details on the suggested signatures for named operations.

3.4 Ccs.BusinessLogic.IGetDataRequest

Interface for the request object for the IBusinessEntity's getData and getResultCount methods.

As the IGetDataRequest and IGetDataTableRequest Interface use a number of EXTENT properties, it is recommended that implementers consider providing constructors or helper methods that simplify passing a reasonable amount of values to the array properties on the fly.

```
INTERFACE Ccs.BusinessLogic.IGetDataRequest:

    /*------------------------------------------------------------------------------
        Purpose: Returns the custom parameter object
        Notes:   May be used to provide further instructions or information to the
                 Business Entity while executing the GetData request
    ------------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY CustomParameter AS Progress.Lang.Object NO-UNDO
    GET.

    /*------------------------------------------------------------------------------
        Purpose: Returns the named query instance
        Notes:
    ------------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY NamedQuery AS INamedQuery NO-UNDO
    GET.

    /*------------------------------------------------------------------------------
        Purpose: Returns the Table requests
        Notes:
    ------------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY TableRequests AS IGetDataTableRequest EXTENT NO-UNDO
    GET.

END INTERFACE.
```

3.4.1 Public instance properties

Name	CustomParameter
Description	A custom parameter object to be used by the Business Entity. A consumer might for instance use this to indicate if an expensive calculated field should be populated or not.
Type	Progress.Lang.Object
Getter	Public
Setter	Not defined
Unknown	This property value might be unknown. Indicating that no custom instructions are provided by the consumer.

Name	NamedQuery
Description	An INamedQuery instance providing a named query to the Business Entity. Named Queries can be combined with TableRequests. But this is not mandatory. A Business Entity might completely ignore the TableRequests when a known named query is references. A Business Entity is expected to throw an error with a meaningful error message, when the consumer provides a reference to an unknown named query or does provide unknown parameters to a named query
Type	INamedQuery
Getter	Public
Setter	Not defined
Unknown	This property value might be unknown. Indicating that no named query is invoked by the consumer.

Name	TableRequests
Description	A array of ITableRequest instances describing the Tables and their selection criteria of this request
Type	ITableRequest EXTENT
Getter	Public
Setter	Not defined
Unknown	The extent of this property value might be unknown. Indicating that no ITableRequest’s are provided to the getData or getResultCount method. When no NamedQuery is provided the Business Entity is expected to return a default response. When the extent of this property is greater than zero, all values MUST be valid ITableRequest instances.

3.5 Ccs.BusinessLogic.IGetDataTableRequest

Interface for the TableRequest property of the request object for the IBusinessEntity's getData and getResultCount methods.

As the IGetDataRequest and IGetDataTableRequest Interface use a number of EXTENT properties, it is recommended that implementers consider providing constructors or helper methods that simplify passing a reasonable amount of values to the array properties on the fly.

```
USING Ccs.BusinessLogic.* FROM PROPATH .
USING Progress.Lang.*     FROM PROPATH .

INTERFACE Ccs.BusinessLogic.IGetDataTableRequest:

    /*------------------------------------------------------------------------------
        Purpose: Returns the paging context
        Notes:   Used for Paging. This value typically consists of record
                 identifiers
                 (e.g. DATA-SOURCE ROWID retrieved by the RESTART-ROWID function
                 Of the previous call into IBusinessEntity:GetData or other data
                 Required by the Business Entity to build the next batch of data).
                 The value passed in is the value of the NextBatchingContext
                 Property of the IGetDataTableResponse for the table
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY PagingContext AS CHARACTER NO-UNDO
    GET.

    /*------------------------------------------------------------------------------
        Purpose: Returns the number of records requested by the caller of the
                 Business Entity getData method
        Notes:   Used for Paging. When the value is 0, the business
                 entity is expected to return all (remaining) records. When the
                 value is ? the business entity is expected to return a reasonable
                 default number of records to the caller. Negative values indicate
                 paging in backwards direction is requested.
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY NumRecords AS INT64 NO-UNDO
    GET.

    /*------------------------------------------------------------------------------
        Purpose: Returns the abstract query defintion for this request
        Notes:   Typically used as an alternative to the QueryString
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY QueryDefinition AS IQueryDefinition NO-UNDO
    GET.

    /*------------------------------------------------------------------------------
        Purpose: Returns the Query String for this table
        Notes:   Query Strings must be expressed using the fields of the temp-
                 table. It’s the task of the Business Entity or Data Access class
                 to translate the Query String into the form understood by the
                 actual DBMS in case field names require mapping etc.
                 Query Strings must be provided in the following format
                 CustNum = 42
                 CustNum = 42 AND OrderStatus = “Ordered”
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY QueryString AS CHARACTER NO-UNDO
    GET.

    /*------------------------------------------------------------------------------
        Purpose: Returns the number of records to skip
        Notes:   Used for Paging. Typically the value of (page# - 1) * NumRecords
                 Is passed in when requesting a certain page of result records
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY Skip AS INT64 NO-UNDO
    GET.
    /*------------------------------------------------------------------------------
        Purpose: Returns the name of the ProDataset Table
        Notes:   Identifies the table this IGetDataTableRequest belongs to
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY TableName AS CHARACTER NO-UNDO
    GET.

END INTERFACE.
```

3.5.1 Public instance properties

Name	PagingContext
Description	When the consumer provides a value for the PagingContext the Business Entity is expected to return the resulting records relatively to this PagingContext. Typically, for the PagingContext the value of the NextPagingContext or PreviousPagingContext of the previous getData() call is provided.
Type	CHARACTER
Getter	Public
Setter	Not defined
Unknown	This property value might be unknown or empty. Indicating that the first set matching the selection criteria is requested.

Name	NumRecords
Description	Defines the number of records requested by the caller of the Business Entity getData() method, used for Paging. When the value is 0, the business entity is expected to return all (remaining) records. When the value is ? the business entity is expected to return a reasonable default number of records to the caller. Negative values indicate that backwards paging is requested.
Type	INT64
Getter	Public
Setter	Not defined
Unknown	This property value might be unknown. See Description for details.

Name	QueryDefinition
Description	Provides an abstract query definition consisting of a list of potentially nested query predicates and a list of sort criteria. The QueryDefinition must be used as an alternative to the QueryString property.
Type	IQueryDefinition
Getter	Public
Setter	Not defined
Unknown	This property value might be unknown. Indicating that no abstract query definition is provided for the request.

Name	QueryString
Description	Provides an ABL query string to the Business Entity getData() request. The query string must be expressed using the field names of the ProDataset temp-table (with no table prefix). The QueryString must be used as an alternative to the QueryDefinition property.
Type	CHARACTER
Getter	Public
Setter	Not defined
Unknown	This property value might be unknown. Indicating that no query string is provided by the consumer.

Name	Skip
Description	Returns the number of records to skip with this request. Used for Paging as an alternative to the PagingContext. Typically the value of (page# - 1) * NumRecords is passed in when requesting a certain page of result records
Type	INT64
Getter	Public
Setter	Not defined
Unknown	This property value might be unknown or zero. Indicating that no records should be skipped or the PagingContext is set.

Name	TableName
Description	Identifies the table this IGetDataTableRequest belongs to. This must be a valid name of a ProDataset member table.
Type	CHARACTER
Getter	Public
Setter	Not defined
Unknown	This property value must not be unknown or empty.

3.6 Ccs.BusinessLogic.IQueryDefinition

The Ccs.BusinessLogic.IQueryDefinition interface provides the foundation of an abstract query definition.

```
USING Ccs.BusinessLogic.* FROM PROPATH .
USING Progress.Lang.*     FROM PROPATH .

INTERFACE Ccs.BusinessLogic.IQueryDefinition:

    /*------------------------------------------------------------------------------
        Purpose: Returns the list of query predicates or query groups for this
                 Query definition
        Notes:
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY QuerySelection AS IQueryEntry NO-UNDO
    GET.

    /*------------------------------------------------------------------------------
        Purpose: Returns the list of query sort entries
        Notes:
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY QuerySort AS IQuerySortEntry EXTENT NO-UNDO
    GET.

END INTERFACE .
```

3.6.1 Public instance properties

Name	QuerySelection
Description	Provides the reference to the query selelection as an IQueryEntry reference. The Query Selection will reference either an IQueryGroup instance or a single IQueryPredicate
Type	IQueryEntry (common base type of IQueryGroup and IQueryPredicate)
Getter	Public
Setter	Not defined
Unknown	This property value might be unknown or empty. Indicating that the consumer provides no query selection but only query sort criteria.

Name	QuerySort
Description	Returns the list of query sort entries requested by the consumer
Type	IQuerySortEntry EXTENT
Getter	Public

Setter	Not defined
Unknown	This property value might be unknown. Indicating that the consumer provides no query sort but only query selection criteria.

3.7 Ccs.BusinessLogic.IQueryEntry

Common base type for IQueryGroup and IQueryPredicate.

```
USING Ccs.BusinessLogic.* FROM PROPATH .
USING Progress.Lang.*     FROM PROPATH .

INTERFACE Ccs.BusinessLogic.IQueryEntry:

    /*----------------------------------------------------------------------------
        Purpose: Returns the logical operator that shold be used to join this
                 query entry to its predecessor in the current list
        Notes:   The value of None is only supported for the first entry
    --------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY Join AS JoinEnum NO-UNDO
    GET.

END INTERFACE .
```

3.7.1 Public instance properties

Name	Join
Description	Provides the reference to the Join criteria (logical operator) required to join the current query entry to its predecessor in a IQueryGroup. The values of None or Not MAY only be used for the first entry in a list.
Type	JoinEnum
Getter	Public
Setter	Not defined
Unknown	This property value must not be unknown.

3.8 Ccs.BusinessLogic.IQueryGroup

Inherits Ccs.BusinessLogic.IQueryEntry.

Represents a list of query predicates or nested query groups.

```
USING Ccs.BusinessLogic.* FROM PROPATH .
USING Progress.Lang.*     FROM PROPATH.

INTERFACE Ccs.BusinessLogic.IQueryGroup
    INHERITS IQueryEntry:

    /*----------------------------------------------------------------------------
        Purpose: Returns the array of query predicates and nested query groups
        Notes:
    ------------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY Entries AS IQueryEntry EXTENT NO-UNDO
    GET.

END INTERFACE.
```

3.8.1 Public instance properties

Name	Entries
Description	Provides the list of query entries represented by this IQueryGroup. Every query entry MAY either by an IQueryPredicate or a nested IQueryGroup. The Join property of each query entry provides the Boolean operator to use between an entry and its predecessor in the list. The Join values of None or Not are only allows for the first entry in the list.
Type	IQueryEntry EXTENT
Getter	Public
Setter	Not defined
Unknown	This property value MUST not be unknown. The array must contain at least two entries (when only one criteria is required, an IQueryPredicate should be used). Every element in the array must be a valid object reference.

3.9 Ccs.BusinessLogic.IQueryPredicate

Inherits Ccs.BusinessLogic.IQueryEntry.

Represents a single query criteria for a single field.

```
USING Ccs.BusinessLogic.*  FROM PROPATH .
USING Ccs.Common.Support.* FROM PROPATH .
USING Progress.Lang.*      FROM PROPATH .

INTERFACE Ccs.BusinessLogic.IQueryPredicate
    INHERITS IQueryEntry:

    /*------------------------------------------------------------------------------
        Purpose: Returns the name of the field for this query predicate
        Notes:
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY FieldName AS CHARACTER NO-UNDO
    GET.

    /*------------------------------------------------------------------------------
        Purpose: Returns the operator for this query predicate
        Notes:
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY Operator AS QueryOperatorEnum NO-UNDO
    GET.

    /*------------------------------------------------------------------------------
        Purpose: Returns a single value for this query predicate
        Notes:
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY Value AS IPrimitiveHolder NO-UNDO
    GET.

    /*------------------------------------------------------------------------------
        Purpose: Returns a list of values for this query predicate
        Notes:   Used by the InRange and InList operators
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY Values AS IPrimitiveArrayHolder NO-UNDO
    GET.

END INTERFACE.
```

3.9.1 Public instance properties

Name	FieldName
Description	Returns the field name this IQueryPredicate provides selection criteria for.
Type	CHARACTER
Getter	Public
Setter	Not defined

Unknown	This property value must not be unknown. It must provide a value temp-table field name.

Name	Operator
Description	Returns the Operator for this IQueryPredicate
Type	QueryOperatorEnum
Getter	Public
Setter	Not defined
Unknown	This property value must not be unknown.

Name	Value
Description	Returns a single value for this query predicate. The Value cannot be used for InRange or InValue operators. The primitive holders are used as an alternative to an “ANY-TYPE” parameter type.
Type	Ccs.Common.Support.IPrimitiveHolder
Getter	Public
Setter	Not defined
Unknown	This property value might be unknown for the InRange or InValue operator. All other operators require the use of a Value.

Name	Values
Description	Returns a list of values for this query predicate. Used by the InRange and InList operators. The primitive array holders are used as an alternative to an “ANY-TYPE” parameter type.
Type	Ccs.Common.Support.IPrimitiveArrayHolder
Getter	Public
Setter	Not defined
Unknown	This property value might be unknown except for the InRange or InValue operator.

3.10 Ccs.BusinessLogic.IQuerySortEntry

Describes a single sort criteria for an abstract query definition.

```
USING Ccs.BusinessLogic.* FROM PROPATH .
USING Progress.Lang.*     FROM PROPATH .

INTERFACE Ccs.BusinessLogic.IQuerySortEntry:

    /*------------------------------------------------------------------------------
        Purpose: Returns the name of the field for this query sort entry
        Notes:
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY FieldName AS CHARACTER NO-UNDO
    GET.

    /*------------------------------------------------------------------------------
        Purpose: Returns the sort order for this query sort entry
        Notes:
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY SortOrder AS SortOrderEnum NO-UNDO
    GET.

END INTERFACE .
```

3.10.1 Public instance properties

Name	FieldName
Description	Returns the field name this IQuerySortEntry instance provides sort criteria for.
Type	CHARACTER
Getter	Public
Setter	Not defined
Unknown	This property value must not be unknown. It must provide a value temp-table field name.

Name	SortOrder
Description	Returns the sort order for this IQuerySortEntry
Type	SortOrderEnum
Getter	Public
Setter	Not defined
Unknown	This property value must not be unknown.

3.11 Ccs.BusinessLogic.JoinEnum

Non-Flagged enumeration of boolean operators used to join multiple query entries in a list (query group). Used for the Join property of the IQueryEntry interface.

```
USING Ccs.BusinessLogic.* FROM PROPATH .
USING Progress.Lang.*     FROM PROPATH .

ENUM Ccs.BusinessLogic.JoinEnum:

    DEFINE ENUM None    /* For the first entry in a list */
                Not     /* For the first entry in a list */
                And
                AndNot
                Or
                OrNot
                .
END ENUM .
```

3.11.1 Enum members

None	Applicable to the first entry in a list only. Indicates that no negation of the entry is required.
Not	Applicable to the first entry in a list only. Indicates that boolean negation of the entry is required.
Ant	Applicable from the second entry in a list on only. Indicates that the boolean AND operator is required.
AntNot	Applicable from the second entry in a list on only. Indicates that the boolean AND NOT operator is required.
Or	Applicable from the second entry in a list on only. Indicates that the boolean OR operator is required.
OrNot	Applicable from the second entry in a list on only. Indicates that the boolean OR NOT operator is required.

3.12 Ccs.BusinessLogic.QueryOperatorEnum

Non-Flagged enumeration of query operators used to specify an IQueryPredicate instance.

```
USING Ccs.BusinessLogic.* FROM PROPATH .
USING Progress.Lang.*     FROM PROPATH .

ENUM Ccs.BusinessLogic.QueryOperatorEnum:

    DEFINE ENUM /* Operators requiring a single value */
                Eq
                Begins
                Contains
                Matches
                Ge
                Gt
                Le
                Lt

                /* Operators requiring a list of values */
                InList
                InRange
                .
END ENUM .
```

3.12.1 Enum members

Eq	Perform an equality match with the primitive value referenced by the Value property.
Begins	Perform a begins match with the primitive value referenced by the Value property.
Contains	Perform a contains match with the primitive value referenced by the Value property (typically requires a word index on the RDBMS).
Matches	Perform a matches match with the primitive value referenced by the Value property.
Ge	Perform a greater or equal match with the primitive value referenced by the Value property.
Gt	Perform a greater than match with the primitive value referenced by the Value property.
Le	Perform a less or equal match with the primitive value referenced by the Value property.
Lt	Perform a less than match with the primitive value referenced by the Value property.

InList	Perform a InList match with the list of primitive values referenced by the Values property.
InRange	Perform a InRange match with the list of primitive values (typically exactly two) referenced by the Values property.

3.13 Ccs.BusinessLogic.SortOrderEnum

Non-Flagged enumeration of sort order used to specify an IQuerySortEntry instance.

```
USING Ccs.BusinessLogic.* FROM PROPATH .
USING Progress.Lang.*     FROM PROPATH .

ENUM Ccs.BusinessLogic.SortOrderEnum:

    DEFINE ENUM Ascending
                Descending
                .
END ENUM .
```

Ascending	The field should be sorted ascending.
Descending	The field should be sorted descending.

3.14 Ccs.BusinessLogic.INamedQuery

Describes a named query request. Used for the NamedQuery property of the IGetDataRequest object.

```
USING Ccs.BusinessLogic.* FROM PROPATH .
USING Progress.Lang.*     FROM PROPATH .

INTERFACE Ccs.BusinessLogic.INamedQuery:

    /*------------------------------------------------------------------------------
        Purpose: Returns the name of the named query
        Notes:
    ------------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY Name AS CHARACTER NO-UNDO
    GET.

    /*------------------------------------------------------------------------------
        Purpose: Returns the array of (optional) parameters of the named query
        Notes:   Each Named Query Parameter consists of an identifier (name) and a
                 value (primitive holder) or values (primitive array holder)
    ------------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY Parameters AS INamedQueryParameter EXTENT NO-UNDO
    GET.

END INTERFACE.
```

3.14.1 Public instance properties

Name	Name
Description	Returns the name of the requested named query
Type	CHARACTER
Getter	Public
Setter	Not defined
Unknown	This property value must not be unknown. It must provide a known name of a named query supported by the Business Entity

Name	Parameters
Description	Returns the list of parameters for this INamedQuery instance
Type	INamedQueryParameter EXTENT
Getter	Public
Setter	Not defined

Unknown	This property value MAY be unknown (EXTENT = ?). When an extent size greater than zero is provided every entry must be a valid object reference.

3.15 Ccs.BusinessLogic.INamedQueryParameter

Describes an individual parameter of an INamedQuery.

```
USING Ccs.BusinessLogic.*  FROM PROPATH .
USING Ccs.Common.Support.* FROM PROPATH .
USING Progress.Lang.*      FROM PROPATH .

INTERFACE Ccs.BusinessLogic.INamedQueryParameter:

    /*------------------------------------------------------------------------------
        Purpose: Returns the name of the named query parameter
        Notes:
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY Name AS CHARACTER NO-UNDO
    GET.

    /*------------------------------------------------------------------------------
        Purpose: Returns a single value for this named query parameter
        Notes:
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY Value AS IPrimitiveHolder NO-UNDO
    GET.

    /*------------------------------------------------------------------------------
        Purpose: Returns a list of values for this named query parameter
        Notes:
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY Values AS IPrimitiveArrayHolder NO-UNDO
    GET.

END INTERFACE.
```

3.15.1 Public instance properties

Name	Name
Description	Returns the name of the named query parameter
Type	CHARACTER
Getter	Public
Setter	Not defined
Unknown	This property value must not be unknown. It must provide a known parameter name for a parameter of a named query supported by the Business Entity

Name	Value
Description	Returns a single value for this named parameter. The primitive holders are used as an alternative to an “ANY-TYPE” parameter type.
Type	Ccs.Common.Support.IPrimitiveHolder
Getter	Public
Setter	Not defined
Unknown	This property value might be unknown when the Values property is not unknown.

Name	Values
Description	Returns a list of values for this named parameter. The primitive array holders are used as an alternative to an “ANY-TYPE” parameter type.
Type	Ccs.Common.Support.IPrimitiveArrayHolder
Getter	Public
Setter	Not defined
Unknown	This property value might be unknown except when the Value property is not.

3.16 Ccs.BusinessLogic.IGetDataResponse

Interface for the response of the getData() method of the IBusinessEntity.

```
USING Ccs.BusinessLogic.* FROM PROPATH .
USING Progress.Lang.*     FROM PROPATH .

INTERFACE Ccs.BusinessLogic.IGetDataResponse:

    /*------------------------------------------------------------------------------
        Purpose: Returns the custom response object
        Notes:   May be used to return further information to the caller. May
                 Return the reference to the IGetDataRequest:CustomParameter
                 object
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY CustomResponse AS Progress.Lang.Object NO-UNDO
    GET.

    /*------------------------------------------------------------------------------
        Purpose: Returns the Table requests
        Notes:
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY TableResponses AS IGetDataTableResponse EXTENT NO-UNDO
    GET.

END INTERFACE.
```

3.16.1 Public instance properties

Name	CustomResponse
Description	Returns the reference to the custom response object. MAY be used to return further arbitrary information to the caller.
Type	Progress.Lang.Object
Getter	Public
Setter	Not defined
Unknown	This property value MAY be unknown.

Name	TableResponses
Description	Returns the array of IGetDataTableResponse instances describing the paging context of every requested temp-table.
Type	IGetDataTableResponse EXTENT
Getter	Public
Setter	Not defined

Unknown	This property value must have an extent size greater than zero. Each array element must contain a valid IGetDataTableResponse reference.

3.17 Ccs.BusinessLogic.IGetDataTableResponse

Interface describing a single entry of the TableResponses property of the IGetDataResponse interface.

```
USING Ccs.BusinessLogic.* FROM PROPATH .
USING Progress.Lang.*     FROM PROPATH .

INTERFACE Ccs.BusinessLogic.IGetDataTableResponse:

    /*------------------------------------------------------------------------------
        Purpose: Returns the paging context to be passed back to the business
                 entity when requesting the next set
        Notes:   Used for Paging. This value typically consists of record
                 Identifiers (e.g. DATA-SOURCE ROWID retrieved by the RESTART-
                 ROWID function or other data required by the Business Entity to
                 build the next set of data in a follow up call).
    ------------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY NextPagingContext AS CHARACTER NO-UNDO
    GET.

    /*------------------------------------------------------------------------------
        Purpose: Returns the paging context to be passed back to the business
                 entity when requesting the previous set
        Notes:   Used for Paging. This value typically consists of record
                 identifiers
                 (e.g. DATA-SOURCE ROWID retrieved by the RESTART-ROWID function
                 or other data required by the Business Entity to build the
                 previous set of data in a follow up call).
    ------------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY PreviousPagingContext AS CHARACTER NO-UNDO
    GET.

    /*------------------------------------------------------------------------------
        Purpose: Returns the name of the ProDataset Table
        Notes:   Identifies the table this IGetDataTableResponse belongs to
    ------------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY TableName AS CHARACTER NO-UNDO
    GET.

END INTERFACE.
```

3.17.1 Public instance properties

Name	NextpagingContext
Description	Returns the paging context to be passed back to the business entity when requesting the next set of records. This value typically consists of record identifiers (e.g. DATA-SOURCE ROWID retrieved by the RESTART-ROWID function or other data required by the Business Entity to build the next set of data in a follow up call).
Type	CHARACTER
Getter	Public
Setter	Not defined
Unknown	This property value MAY be unknown or empty indicating that no next set of data is present.

Name	PreviousPagingContext
Description	Returns the peging context to be passed back to the business entity when requesting the previous set. This value typically consists of record identifiers (e.g. DATA-SOURCE ROWID retrieved by the RESTART-ROWID function or other data required by the Business Entity to build the previous set of data in a follow up call).
Type	CHARACTER
Getter	Public
Setter	Not defined
Unknown	This property value MAY be unknown or empty indicating that no previous set of data is present.

Name	TableName
Description	Identifies the table this IGetDataTableResponse belongs to. This must be a valid name of a ProDataset member table.
Type	Progress.Lang.Object
Getter	Public
Setter	Not defined
Unknown	This property value must not be unknown or empty.

3.18 Ccs.BusinessLogic.IGetResultCountResponse

Interface for the response object of the getResultCount() method of the Business Entity interface.

```
USING Ccs.BusinessLogic.* FROM PROPATH .
USING Progress.Lang.*     FROM PROPATH .

INTERFACE Ccs.BusinessLogic.IGetResultCountResponse:

    /*------------------------------------------------------------------------------
        Purpose: Returns the result counts per requested table
        Notes:
    ------------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY ResultCounts AS IGetTableResultCountResponse EXTENT
        NO-UNDO
    GET.

END INTERFACE.
```

3.18.1 Public instance properties

Name	ResultCounts
Description	Returns the array of result counts per requested table.
Type	IGetTableResultCountResponse EXTENT
Getter	Public
Setter	Not defined
Unknown	This property value must have an extent value greater than zero. Every entry in the array must be a reference to a valid IGetTableResultCountResponse..

3.19 Ccs.BusinessLogic.IGetTableResultCountResponse

Interface for the IGetResultCount portion for a single temp-table represented by the ResultCounts array of the IGetResultCountResponse.

```
USING Ccs.BusinessLogic.* FROM PROPATH .
USING Progress.Lang.*     FROM PROPATH .

INTERFACE Ccs.BusinessLogic.IGetTableResultCountResponse:

    /*------------------------------------------------------------------------------
        Purpose: Returns is the result is exact (TRUE) or Guessed or Cached 
(FALSE)
        Notes:
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY Exact AS LOGICAL NO-UNDO
    GET.

    /*------------------------------------------------------------------------------
        Purpose: Returns the number of results for this table
        Notes:
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY NumResults AS INT64 NO-UNDO
    GET.

    /*------------------------------------------------------------------------------
        Purpose: Returns the name of the table this result belongs to
        Notes:
    ----------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY TableName AS CHARACTER NO-UNDO
    GET.

END INTERFACE.
```

3.19.1 Public instance properties

Name	Exact
Description	Returns true when the NumResults property value in this IGetTableResultCountResponse instance was the result of an exact counting and false when the NumResults property is a guessed or cached value.
Type	LOGICAL
Getter	Public
Setter	Not defined
Unknown	This property value must not be the unknown value.

Name	NumResults
Description	Returns the number of records this IGetTableResultCountResponse represents.
Type	INT64
Getter	Public
Setter	Not defined
Unknown	This property value MAY be the unknown value indicating that the Business Entity is not able to provide a result in any reasonable way.

Name	TableName
Description	Identifies the table this IGetTableResultCountResponse belongs to. This must be a valid name of a ProDataset member table.
Type	CHARACTER
Getter	Public
Setter	Not defined
Unknown	This property value must not be unknown or empty.

3.20 Ccs.BusinessLogic.IUpdateDataRequest

Interface for the request object for the updataData() method of the updatable Business Entity.

```
USING Ccs.BusinessLogic.* FROM PROPATH .
USING Progress.Lang.*     FROM PROPATH .

INTERFACE Ccs.BusinessLogic.IUpdateDataRequest:

    /*------------------------------------------------------------------------------
        Purpose: Returns the CommitScope to be used by the updateData method
        Notes:   The value is considered as a recommendation as the Business
                 Entity may ignore this setting and use a different commit scope
                 based on the business logic requirements
    ------------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY CommitScope AS CommitScopeEnum NO-UNDO
    GET.


    /*------------------------------------------------------------------------------
        Purpose: Returns a custom request object
        Notes:
    ------------------------------------------------------------------------------*/
    DEFINE PUBLIC PROPERTY CustomRequest AS Progress.Lang.Object NO-UNDO
    GET.

END INTERFACE.
```

3.20.1 Public instance properties

Name	CommitScope
Description	Returns the CommitScope to be used by the updateData() method. The value is considered as a recommendation as the Business Entity MAY ignore this setting and use a different commit scope based on the business logic requirements
Type	CommitScopeEnum
Getter	Public
Setter	Not defined
Unknown	This property value MAY be the unknown value indicating that the consumer provides no hint on the commit scope to the Business Entity.

Name	CustomRequest
Description	Returns the custom parameter object provided to the updateData() method. It’s up to the implementer of the Business Entity to define the meaning of this parameter.
Type	Progress.Lang.Object
Getter	Public
Setter	Not defined
Unknown	This property value MAY be the unknown value.

3.21 Ccs.BusinessLogic.CommitScopeEnum

Enumeration for the CommitScope property of the IUpdataDataRequest interface.

```
USING Ccs.BusinessLogic.* FROM PROPATH .
USING Progress.Lang.*     FROM PROPATH .

ENUM Ccs.BusinessLogic.CommitScopeEnum:

    DEFINE ENUM /* All records in the dataset in a single tx */
                All

                /* One transaction per table row */
                Row

                /* One transaction for all records in a single table */
                Table

                /* One database transaction per parent row and its child table
                   and all grand-child records. */
                Nested

                Default = Row
                .
END ENUM .
```

3.21.1 Enum members

All	All records in the dataset in a single transaction.
Row (Default)	One transaction per table row.
Table	One transaction for all records in a single table.
Nested	One database transaction per parent row and its child table and all grand-child records.

4 Guidelines for implementers

It is mandatory that the Business Entity is developed with no particular type of consumer in mind. Any data conversion required to support a particular consumer must be implemented in a Service Interface component – outside of the Business Entity.

The Business Entity leverages the ProDataset before-image feature for updating data. Only a single updateData() method is provided supporting all three types of data manilpulations: Create, update and delete. It’s the job of the Service Interface to provide specializes entry points for clients that do not support ProDatasets with before-image support.

A Business Entity is a critical component for the runtime performance of a business application. Especially business entity read operations are required to be implemented with optimal performance in mind. A Business Entity is responsible for deciding if a certain request is acceptable to be executed and throw an error for requests that are not, e.g. when critical components of an index are not provided during a read operation of a large database table.

During the implementation of the Business Entity it is important to ensure that the execution of the getResultCount() method does not result in a poor system performance as this method MAY be executed fairly often by consumers.

Document Control

Title:	OERA Business Entity Standard
Version:	1.0

Document History

Date	Version	Author	Change Details
27/05/2015	**0.1**	**Mike Fechner**	**Initial revision**
14/06/2016	**0.3**	**Mike Fechner**	**Changes after review by spec team**
16/06/2016	**0.4**	**Mike Fechner**	**Changes after review by spec team**
17/06/2016	**0.5**	**Mike Fechner**	**Changes after review by spec team**
21/06/2016	**1.0**	**Mike Fechner**	**Version released to Community Review**

5 Outstanding Issues

- Request authorization: When the Service Interface component will be discussed by the CCS group we have to make a decision on who’s responsible for implementing request authorization: The Business Entity or the Service Interface.

  Implementing this in the Service Interface simplifies to keep this out of the scope of the business logic and facilitates generic and reusable solutions for this purpose.

  However, this requires that all calls from a business entity into another business entity (potentially crossing application domain boundaries that require proper authorization) would have to go through the service interface as well. Otherwise we cannot prevent that a business entity that is unrestricted for a certain consumer would be able to call on behalf of that consumer into a business entity that the original consumer would not have access to.