Common Component Specification
A Deep Dive

Mike Fechner, Consultingwerk Ltd.
mike.fechner@consultingwerk.de
Consultingwerk Ltd.

- Independent IT consulting organization
- Focusing on OpenEdge and related technology
- Located in Cologne, Germany
- Customers in Europe, North America, Australia and South Africa
- Vendor of tools and consulting programs
- 26 years of Progress experience (V5 … OE11)
- Specialized in GUI for .NET, OO, Software Architecture, Application Integration

http://www.consultingwerk.de/
Agenda

- CCS
  - Disclaimers
  - Spec status
  - ABL Version Dependency
  - Services and Managers
  - Startup Manager
  - Service Manager
  - Session Manager
  - Business Entities
  - Conclusion
What is the CCS

- Common Component Specification (CCS) project is a mechanism for developing standard business application component specifications for business applications
- Driven by OpenEdge experts and evangelists from Progress Community
- OpenEdge community members can participate
Advantages of the CCS

- Versioned group of CCS specs can be used by develop a compliant application framework
- Enables creation of **standards** based framework components that can interoperate
- Enables creation of standard tools for framework components
- Fosters a vibrant community of Service Delivery Partners that can **collaborate** and compete
Advantages of the CCS

- Pick the best component for the job
  - Plug and play based on defined standards
- Vendor A’s ERP Business Entities with Vendor B’s CRM Business Entities and Vendor C’s Service Manager and a customized Authorization Manager
- Avoid vendor lock in
- Common understanding of framework architecture
- Developers should understand different frameworks easier
Elements of CCS

- Community Forum
  https://community.progress.com/products/directions/common_component/

- Steering Committee – provides governance (Progress Dev, Progress Bravepoint, Large AP, Large Direct EU, Consulting Partner)

- Multiple spec teams

- Legal documents (spec license, publisher agreement)

- NDA removed on popular demand
CCS Deliveries

- Specification (PDF Document)
- Interface definitions (Source code)
- Optionally sample code or test cases
- PDF Document and Interface must be followed by implementers that claim compatibility
- NO reference implementation!!!

- Current expectation is that CCS delivers only a single **Class** and many **Interfaces** and **Enum**’s as part of the specifications
How does CCS work

- Spec idea proposed to the steering committee
- Spec idea is approved, spec team is founded (min of 3 people from 3 different companies), submitter of the idea typically team-leader
- Typically weekly conf-calls in spec team (Lync/Skype for Business, etc.)
- Spec draft gets written by small group
- Community review after approval by the steering committee
- Public release
Agenda

- CCS

- **Disclaimers**
  - Spec status
  - ABL Version Dependency
  - Services and Managers
  - Startup Manager
  - Service Manager
  - Session Manager
  - Business Entities
  - Conclusion
Disclaimer 1

- Specs are **not yet published**. Some specs are in community review, other specs are still in development.
- What you’ll see in this presentation is a **best guess** of the final Interface definitions and requirements for the components.
- Everything is subject to change without notion – until specs are finally published to the public!
Disclaimer 2

- Sample code not (yet) publicly available
- Sample code shown will be published as the spec status progresses
  - Community review
  - Final release of spec
Disclaimer 3

- Aim of the CCS is to define specifications and interfaces
- Sample code will be delivered with no support at all
- Sample code may be incomplete
- CCS is **NOT aiming to collaboratively build an open-source framework** and the sample code is not the starting point for that
- However, other organizations may use the code as a starting point for an open-source framework
Agenda

- CCS
- Disclaimers
- Spec status
- ABL Version Dependency
- Services and Managers
- Startup Manager
- Service Manager
- Session Manager
- Business Entities
- Conclusion
The OpenEdge Application Architecture (OEAA)

- Adheres to the OpenEdge Reference Architecture (OERA)
- Defines naming standards for namespaces, interfaces, classes, methods and properties
- Defines the managers that make up the Common Infrastructure
- Required and recommended components
The OpenEdge Application Architecture (OEAA)

▪ Common Components
  – Startup Manager *
  – Session Manager *
  – Service Manager *
  – Connection Manager
  – Property Manager
  – Context Data Manager
  – Authorization Manager
  – Authentication Manager
  – Messaging Manager
  – Logging Manager
  – Translation Manager
  – Analytics Manager

* Required Component in Version 1 of CCS

▪ Business Services
  • Business Entity
  • Workflow
  • Task

▪ Data Access
  • Database or Data Servers: Open or SQL
  • Data Synchronization DB

▪ Service Interfaces
  • Data Service Catalog
  • Service API
  • UI Metadata

▪ Presentation
  • Client Data Object
Spec status

- **CCS-ARCH01**: Overall architecture spec, released February 2016

- **CCS-SSS**:  
  - Startup Manager: Community Review  
  - Service Manager: Draft under development  
  - Session Manager: Draft under development

- **CCS-BE**: Business Entity: Community Review
Nuff said .... let's talk about code
Agenda

- CCS
- Disclaimers
- Spec status
- **ABL Version Dependency**
  - Services and Managers
  - Startup Manager
  - Service Manager
  - Session Manager
  - Business Entities
- Conclusion
ABL/OpenEdge Version Dependency

- Conscious decision to use latest ABL/OpenEdge language features (currently 11.6)
  - Enums (OE11.6)
  - Interface Inheritance (OE11)
- Community member decision (Progress was more conservative on that subject)
- CCS is for modernization
- Modernization based only on “historic” language features demotivating for spec team members
ABL/OpenEdge Version Dependency

- Enums are made for precise interface definitions
- CommitScopeEnum of “Row, Table, Nested, Dataset“ clearer than integer value of 1..4
- Implementers on older OE releases may use alternative Enum implementations, like https://github.com/consultingwerk/ListsAndEnum Samples (works on 10.2B ... 11.x)

- Interface inheritance reduces need for CAST, simpler TYPE-OF, clearer, ensures that an IUpdatableBusinessEntity is an IBusinessEntity
Agenda

- CCS
- Disclaimers
- Spec status
- ABL Version Dependency

**Services and Managers**
- Startup Manager
- Service Manager
- Session Manager
- Business Entities
- Conclusion
Services and Managers

- A Service is a self-contained unit of functionality
- Well defined functionality and behavior
- Well defined interfaces and life-cycle

- A Manager is a common infrastructure service

- A Business Service is a service implementing business domain specific functionality
Services and Managers

- Typically implementing Interfaces (provided by CCS)
- Requires OO implementation or facade to procedural implementation or remote service
- References accessed through the Service Manager (Services) or Startup Manager (Managers)
Services

- All Services must provide a standard constructor
- All Services must provide an initialize() method

```java
/*
File : IService
Purpose : To be implemented by all Services that are instantiated by the service manager
Syntax :
Description :
Author(s) : rosmith
Created :
Notes :

interface CCS.Common.IService:

    method public void initialize ()..

end interface.
*/
```
Managers

- Same requirements as Services
- Managers are specialized Services

``````
Agenda

- CCS
- Disclaimers
- Spec status
- ABL Version Dependency
- Services and Managers
  - **Startup Manager**
    - Service Manager
    - Session Manager
    - Business Entities
    - Conclusion
Startup Manager

- Applications require a bootstrap process to provide common infrastructure
- During the bootstrap process not all mandatory components may be accessible or initialized yet
- How are you supposed to locate the LoggingManager to log an error while with starting the SessionManager?
- How are you supposed to know which logfile to write startup issues to before a ConfigurationManager can provide that piece of configuration?
Startup Manager

- During the startup process services may need to use alternative functionality when other Managers are not yet available
- Managers must only depend on Managers
- Startup Manager factory for Managers, a special kind of Service
- Startup Manager must provide a means to resolve dependencies, eg. by configuration
Startup Manager

- Startup Manager remains started also after the application/framework launch is completed
- Managers are accessed through the Startup Manager
- Services are accessed through the Service Manager
CCS - A deep dive

Ccs.Common.IStartupManager

/*****************************************************************************
 *                                                                          *
 * File : IStartupManager                                                  *
 * Purpose : The factory of all common components.                         *
 * Syntax :                                                                *
 * Description :                                                          *
 * Author(s) : Simon L Prinsloo                                            *
 * Created : Wed May 18 19:41:55 CAT 2016                                  *
 * Notes :                                                                *
 *******************************************************************************/

USING Ccs.Common.* FROM PROPATH.

INTERFACE Ccs.Common.IStartupManager INHERITS IManager:

/*****************************************************************************
 * Purpose: Retrieve an instance of the specified IManager object.          *
 * Notes:                                                                  *
 @param pServiceType The Progress.Lang.Class representing the required service.  *
 @return IManager implementation of the requested type, or ? if none are configured.   *
*******************************************************************************/

METHOD PUBLIC IManager getManager ( pServiceType AS Progress.Lang.Class ).

END INTERFACE.
Accessing the Startup Manager

- All services accessed through Service Manager
- All managers accessed through Startup Manager (this includes the Service Manager)
- How do we request the reference to the Startup Manager? The one and only?

Singletons are not a useful pattern when the implementation must be exchangeable!
Singleton joins factory, registry and implementation in the actual implementation.
Ccs.Common.Application class

CLASS Ccs.Common.Application FINAL:

    /*
    * Purpose: Provides access to the injected IStartupManager.
    * Notes: 
    */
    DEFINE STATIC PUBLIC PROPERTY StartupManager AS IStartupManager NO-UNDO GET. SET.

    /*
    * Purpose: Provides access to the injected IServiceManager.
    * Notes: 
    */
    DEFINE STATIC PUBLIC PROPERTY ServiceManager AS IServiceManager NO-UNDO GET. SET.

    /*
    * Purpose: Provides access to the injected ISessionManager.
    * Notes: 
    */
    DEFINE STATIC PUBLIC PROPERTY SessionManager AS ISessionManager NO-UNDO GET. SET.

    /*
    * Notes: 
    */
    DEFINE STATIC PUBLIC PROPERTY Version AS CHARACTER NO-UNDO
Application class

- The only Class CCS will ever publish
- Naming it *Framework* seemed wrong …
- Provides access to the Startup Manager
- Provides access to the Service Manager (for convenience)
- Yes, its like a GLOBAL SHARED Variable
- But there does not seem any better way
Sample implementation

- boot.p (or AppServer startup procedure)

```
USING Ccs.Common.* FROM PROPATH.
USING Consultingwerk.CcsSamples.StartupManager.* FROM PROPATH.

/* ***************************************** Main Block ********************************************* */
Application:StartupManager = NEW StartupManager().
Application:StartupManager:initialize().
```

- Creates an instance of a vendor specific StartupManager
- Registers it with the static property of the CCS Application object
Sample Startup Manager

- Configuration provided through startup parameter
  - `param managers=path/to/managers.xml`

- Makes managers exchangeable
  - Mix implementations from different vendors
  - Mockable for unit testing
* Purpose: Initializes the Service
  * Notes: Implements the actual framework bootstrap

METHOD PUBLIC VOID initialize () :

DEFINE VARIABLE cManagersFile AS CHARACTER NO-UNDO INIT "managers.xml":U.
DEFINE VARIABLE i AS INTEGER NO-UNDO.
DEFINE VARIABLE iCount AS INTEGER NO-UNDO.

/* Do we have a managers= entry in -param */
ASSIGN iCount = NUM-ENTRIES (SESSION:PARAM) .

paramLoop:
DO i = 1 TO iCount:
  IF ENTRY (i, SESSION:PARAM) BEGINS "managers=":U THEN DO:
    ASSIGN cManagersFile = ENTRY (2, ENTRY (i, SESSION:PARAMETER), ":=":U).
    LEAVE paramLoop.
  END.
END.

FILE-INFO:FILE-NAME = cManagersFile.

IF FILE-INFO:FULL-PATHNAME = ? THEN
  UNDO, THROW NEW AppError (SUBSTITUTE ("Invalid managers definition: &1", cManagersFile).
THIS-OBJECT:InitializeManagers().

END METHOD.
managers.xml

```xml
<?xml version="1.0"?>
<ttManagers xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ttManagersRow>
    <InitializationSequence>1</InitializationSequence>
    <ManagerType>CCS.Common.IServiceManager</ManagerType>
    <ManagerImplementation>Consultingwerk.CcsSamples.ServiceManager.ServiceManager</ManagerImplementation>
  </ttManagersRow>
</ttManagers>
```
METHOD PROTECTED VOID InitializeManagers():

DEFINE VARIABLE oType AS Progress.Lang.Class NO-UNDO.
DEFINE VARIABLE oInstanceType AS Progress.Lang.Class NO-UNDO.
DEFINE VARIABLE oInstance AS Progress.Lang.Object NO-UNDO.

DEFINE BUFFER ttManagerInstances FOR ttManagerInstances.

FOR EACH ttManagers WHERE ttManagers.InitializationSequence > 0:

ASSIGN
  oType = Progress.Lang.Class:GetClass (ttManagers.ManagerType)
  oInstanceType = Progress.Lang.Class:GetClass (ttManagers.ManagerImplementation)
  oInstance = oInstanceType:GetConstructor (Flags:Public,
                                              NEW ParameterList (0)):Invoke (NEW ParameterList)

IF NOT TYPE-OF (oInstance, IManager) THEN
  UNDO, THROW NEW AppError (SUBSTITUTE ("&1 is not an IManager", oInstanceType:TypeName), 0).

CAST (oInstance, IManager):initialize().

CREATE ttManagerInstances.
ASSIGN ttManagerInstances.ManagerType = oType
  ttManagerInstances.ManagerImplementation = oInstance.

RELEASE ttManagerInstances.

CASE oType:
  WHEN GET-CLASS (IServiceManager) THEN
    Application:ServiceManager = CAST (oInstance, IServiceManager).
  WHEN GET-CLASS (ISessionManager) THEN
    Application:SessionManager = CAST (oInstance, ISessionManager).
  WHEN GET-CLASS (IStartupManager) THEN
    Application:StartupManager = CAST (oInstance, IStartupManager).
END CASE.
Agenda

- CCS
- Disclaimers
- Spec status
- ABL Version Dependency
- Services and Managers
- Startup Manager

- Service Manager
  - Session Manager
  - Business Entities
  - Conclusion
Service Manager

- Service Manager provides access to Services (that are not Managers)
- Factory for business services
- Calls their initialize() method
- Controls their life time
- Services typically launched at first request
- Services may be stopped (at the end of a request, after 1 hour, ...)
Ccs.Common.IServiceManager

using Ccs.Common.*;

interface Ccs.Common.IServiceManager inherits IManager:

    method public IService getService( input poServiceClass as Progress.Lang.Class ).
    method public IService getService( input poServiceClass as Progress.Lang.Class,
                                         input pcInstanceName as character ).

    method public void stopService( input poServiceClass as Progress.Lang.Class,
                                         input pcInstanceName as character ).

end interface.
METHOD PUBLIC VOID initialize ():

DEFINE VARIABLE cServicesFile AS CHARACTER NO-UNDO INIT "services.xml":U.
DEFINE VARIABLE i AS INTEGER NO-UNDO.
DEFINE VARIABLE iCount AS INTEGER NO-UNDO.

/* Do we have a services= entry in -param */
ASSIGN iCount = NUM-ENTRIES (SESSION:PARAM) .

paramLoop:
DO i = 1 TO iCount:
   IF ENTRY (i, SESSION:PARAM) BEGINS "services=":U THEN DO:
      ASSIGN cServicesFile = ENTRY (2, ENTRY (i, SESSION:PARAMETER), ":=":U) .
      LEAVE paramLoop .
   END.
END.

FILE-INFO:FILE-NAME = cServicesFile .

IF FILE-INFO:FULL-PATHNAME = ? THEN
   UNDO, THROW NEW AppError (SUBSTITUTE ("Invalid services definition: &1", cServicesFile) .
I
END METHOD.
METHOD PUBLIC CCS.Common.IService getService (poServiceClass AS Progress.Lang.Class, pcInstanceName AS CHARACTER):

DEFINE VARIABLE oInstanceType AS Progress.Lang.Class NO-UNDO.
DEFINE VARIABLE oInstance AS Progress.Lang.Object NO-UNDO.
DEFINE VARIABLE oError AS AppError NO-UNDO.

DEFINE BUFFER ttServices FOR ttServices.
DEFINE BUFFER ttServiceInstances FOR ttServiceInstances.

// treat ? like empty string - as empty string will cause uniqueness in ttServiceInstances
IF pcInstanceName = ? THEN
    ASSIGN pcInstanceName = "":U.

// existing instance?
IF CAN-FIND (ttServiceInstances WHERE ttServiceInstances.ServiceType = poServiceClass
    AND ttServiceInstances.InstanceName = pcInstanceName) THEN

    FIND ttServiceInstances WHERE ttServiceInstances.ServiceType = poServiceClass
    AND ttServiceInstances.InstanceName = pcInstanceName.

    RETURN CAST (ttServiceInstances.ServiceImplementation, IService).

END.

DO ON ERROR UNDO, THROW:
// Find ttServices record for this request
IF pcInstanceName = "":U THEN DO:
Sample Service Manager implementation

- services.xml file defines service types
- `-param services=path/to/services.xml`

```xml
<ttServices
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <ttServicesRow>
    <ServiceType>Ccs.BusinessLogic.IBusinessEntity</ServiceType>
    <InstanceName>Customer</InstanceName>
  </ttServicesRow>
  <ttServicesRow>
    <ServiceType>Ccs.BusinessLogic.IBusinessEntity</ServiceType>
    <InstanceName>Matches</InstanceName>
  </ttServicesRow>
  <ttServicesRow>
    <ServiceType>Consultingwerk.CcsSamples.CustomerService.IEuro2016ResultService</ServiceType>
    <InstanceName />
  </ttServicesRow>
</ttServices>
```
Accessing a Service

- `Ccs.Common.Application:ServiceManager` reference to the IServiceManager

```plaintext
DEFINE VARIABLE oMatchesService AS IEuro2016ResultService NO-UNDO.

/* ******************************************* Main Block ******************************************* */

// One time bootstrap
RUN Consultingwerk/Ccssamples/boot.p.

oMatchesService = CAST (Application:ServiceManager:get$service
                        (GET-CLASS (IEuro2016ResultService)),
                        IEuro2016ResultService).

oResults = oMatchesService:GetTodaysMatchResults().
```
Demo

- Using the same consumer code to access two different implementation of the same domain service Interface

- Accessing Euro 2016 results from
  - Matches Business Entity
  - Public REST Web Services

- param services=services.xml
- param services=services-online.xml
Agenda

- CCS
- Disclaimers
- Spec status
- ABL Version Dependency
- Services and Managers
- Startup Manager
- Service Manager
- Session Manager
- Business Entities
- Conclusion
Session Manager

- Every request made to a client session is for a particular client session
- Client session has context information associated with it, such as the user and regional settings
- Modern business applications typically state-free
Session Manager

- Validates the authentication token of a session
- Asserts the identity of the client on the session and connected DB’s
- Establishes app server runtime session context data
- Configure ABL session attributes, like regional settings
- Ends a session at the end of the request
Agenda

- CCS
- Disclaimers
- Spec status
- ABL Version Dependency
- Services and Managers
- Startup Manager
- Service Manager
- Session Manager

- Business Entities
- Conclusion
Business Entity

- Standard component for data-access, -update and business logic
  - 3 described ways of issuing queries, support for batching and paging
  - update data
  - invokable methods
- Central element of OERA (John Sadd, et. al.)
- CCSBE does strongly recommend use of a separate data access layer, but does not enforce it (may be enforced later however)
Business Entity

- Interfaces build around a primary ProDataset definition
- Rely on ProDataset before-image (ROW-STATE, ERROR, ERROR-STRING)
- A yet to be defined service interface component is responsible for providing specific interfaces to consumers not capable of consuming or providing ProDatasets that meet this requirement
Business Entities used in PSC Products

- JSDO
- Kendo UI, Kendo UI Mobile
- Kendo UI Builder for OpenEdge
- ADM2, Dynamics
- Rollbase
- DataDirect OpenAccess SDK for OpenEdge
JSDO Compatibility

- CCSBE compatible with JSDO and Kendo UI DataSource
- JFP Pattern
  - ABL like query string
  - top/skip based paging
- count request
  - emphasize the need to provide a well performing implementation, including the ability to return guesses or a cached return
Interface Definitions

- Death by Powerpoint was yesterday
- Today is death by Interfaces …
- 43 Interfaces (17 in Ccs.BusinesLogic package, 26 supporting Primitive Holder Interfaces)
- 4 Enums
Interfaces

- IBusinessEntity, IUpdatableBusinessEntity, ISupportNamedOperations
- IGetDataRequest, IGetDataDataTableRequest, IUpdateDataRequest
- IGetDataResponse, IGetDataTableResponse
- IGetResultCountResponse, IGetTableResultCountResponse
- INamedQuery, INamedQueryParameter
- IQueryDefinition, IQueryEntry, IQueryGroup, IQueryPredicate, IQuerySortEntry
GetData() method

*/

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).
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 PROPERTYNamedQuery AS INamedQuery NO-UNDO GET.

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

INTERFACE IGetDataTableRequest:

Member of Ccs.BusinessLogic
Query by ABL Query String

- GetDataRequest request object as parameter to GetData() method
- GetDataTableRequest sub-structure with data for Query String
- Query String expressed against temp-tables, not DB-tables (consumer can only ask what he knows)
- Support for ABL Query String, excluding
  - FOR EACH
  - WHERE
  - Table Name in Query String
Query by ABL Query String

- Sample constructor of a GetDataRequest implementation
- Provides eCustomer as the table to be requested and ABL Query String

```csharp

oCustomerEntity:getData (NEW GetDataRequest("eCustomer", "SalesRep = 'HXM' AND Name BEGINS 'Li'"), OUTPUT DATASET dsCustomer).
```
Demo

- Demo get-customer-by-querystring.p
Query by abstract query

- Abstracted query expressed based on nested list of objects (arrays for the lack of native IList)
- Reduces ABL knowledge on the side of the consumer
  - no need to teach a JavaScript programmer, that ABL’s “like” is called “matches” or “contains”
- Simplifies Query Optimization in the Business Entity
Query by abstract query

- IQueryEntry: either an IQueryPredicate or an IQueryGroup
- IQueryPredicate: single criteria for a single field
  - Field Name
  - Operator (Enum)
  - Value or Values (InRange, InList)
- IQueryGroup: list (array) of IQueryEntry instances
  - IQueryPredicate or
  - IQueryGroup (supports nesting)
Query by abstract query

- Actual query values are provided through holder classes
  - ICharacterHolder (property Value as CHAR)
  - IIntegerHolder (property Value as INT)
  - ...

- True primitive data types passed to Business Entity - no STRING(anything)

- Bypass ANYTYPE restriction in the ABL
Query by abstract query

- Sample constructor of a GetDataRequest implementation
- Provides eCustomer as the table to be requested and an abstract query
- QueryEntry:Array: Factory for IQueryEntry[]
Demo

- Demo of get-customer-by-abstract-query.p
Named Queries

- Named Queries provide an even better abstraction of the Query
  - OpenOrdersOfCustomer (42)
  - OrdersOfToday
  - SoccerGamesOfToday

- Business Entity solely responsible for providing a query string

- Avoid need to implement similar queries at different consumers, simplify testing
Name Queries

- Query Name
- Query Parameters, pairs of
  - Parameter Name
  - Value (Primitive Holder)
Named Queries

```csharp
oMatches = CAST (Application:ServiceManager:getService(GET-CLASS (IBusinessEntity), "Matches"),
                   IBusinessEntity).

oCount = oMatches:getResultCount (NEW GetDataRequest (NEWNamedQuery("today":U))).

MESSAGE STRING (oMatches) SKIP
   "There are" oCount:ResultCounts[1]:NumResults "matches today."
VIEW-AS ALERT-BOX.

// And now the matches of the world champion
oMatches:getData(NEW GetDataRequest(NEWNamedQuery("team",
                                           "team",
                                           NEW CharacterHolder ("Germany"))), // first

OUTPUT DATASET dsMatches BY-REFERENCE).
```
Demo

- Demo of get-matches-by-named-query.p
Updating data

- Implementation of IUpdatableBusinessEntity optional, not every Business Entity is updatable
- Expects ProDataset with changes (ROW-STATE) as parameter
- Validation is a key business logic requirement
- Validation is NOT expected to THROW errors
- Errors reported within the ProDataset ERROR-STRING property
IUpdatableBusinessEntity

- Extends IBusinessEntity
- Parameter object provides
  - Custom parameter
  - CommitScope

/*
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 not be empty)
@param poUpdateDataRequest The optional request object that allows to provide custom information
@return An optional response object returned by the method
*/

METHOD PUBLIC Progress.Lang.Object updateData (INPUT-OUTPUT DATASET-HANDLE phDataset, poUpdateDataRequest AS IUpdateDataRequest).
Updating Data

- ERROR-STRING used to store a JSON Array of validation errors
  - Field Name (optional)
  - Full text messages (for consumers not able to use a message service)
  - MessageGroup
  - MessageId
  - SubstitutionValues

\{ allows consumer to build localized msg
Demo

- Updating (and validation) in update-customer.p
Agenda

- CCS
- Disclaimers
- Spec status
- ABL Version Dependency
- Services and Managers
- Startup Manager
- Service Manager
- Session Manager
- Business Entities

- Conclusion
Conclusions - Lessons learned

- Time intensive – but fun and very civilized
- A great forum for exchanging and shaping great ideas and experience
- Spec teams may turn in cycles – requiring the team leader to resolve at points
- Community can overrule Progress
- All spec teams so far have exceeded the 90 day period for completing the draft

**Seeking more collaborators!**
Conclusions - Lessons learned

- ABL needs better Array support
  - on the fly passing of Array’s
  - CAST to Class[] type
- ABL needs delegate types
  - Interface for a single method with free choice of method name
Sample code

- Access to sample code will be provided during community review
  - ZIP files attached to forum
  - Access to private Github repository
    https://github.com/consultingwerk/CCS_Samples granted to CCS members on request
Don‘t miss our other presentations

- Monday 11:00: CCS Deep Dive (Mike)
- Tuesday 11:00: OO-Oh (Mike)
- Tuesday 13:00: Application Modernization using the SmartComponent Library (Mike and Marko)
- Tuesday 16:45: REST in Peace (Mike)
- Wednesday 11:00: CCS BoF (all CCS)
- Wednesday 11:00: Angular JS for OpenEdge programmers (Marko)
Questions