CCS - Common Component Specification
A Deep Dive into an implementation

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

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

- CCS
  - Spec status
  - Basic Storage Areas
  - Services and Managers
  - Startup Manager
  - Service 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**.
- Enables creation of **standards** based framework components that can interoperate.
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 end user, 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! – for now …
- Current expectation is that CCS delivers only a single **Class** and many **Interfaces** and **Enum’s** as part of the specifications
Agenda

- CCS
  - Spec status
    - Basic Storage Areas
    - ABL Version Dependency
    - Services and Managers
    - Startup Manager
    - Service Manager
    - Business Entities
    - Conclusion
The OpenEdge Application Architecture (OEAA)
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, released fall 2016  
  - Service Manager, released fall 2016  
  - Session Manager, released fall 2016
- **CCS-BE**: Business Entity, released summer 2016
- **CCS-BSA**: Basic storage areas: Spec draft available
- **CCS-App Metadata**: Work in process (slowly but surely)
A major milestone: CCS is in the box

- OpenEdge 11.7.2 ships with the CCS-SSS and CCS-BE interfaces in the box
Agenda

- CCS
- Spec status
- **Basic Storage Areas**
  - Services and Managers
  - Startup Manager
  - Service Manager
  - Business Entities
- Conclusion
Basic Storage Areas

- Let’s face it: I’m not a DBA. Software architecture and programming are way more interesting than DBA tasks for me.
- Our framework database has started from *prodb SmartDB empty* and remained at that basic structure since then.
- I kept recommending to customers, to optimize the database structure based on their specific requirements before they deploy their application to end users.
- Reality is … no one did. And Tom Bascom got upset with me when he supported one of our clients in production.
Basic Storage Areas

- Goal of the Basic Storage Areas spec is to define best practices to create a simple starting point for creating a new physical database structure
- A good enough starting point for everybody
- Basically defining the new empty.db

- Consequence for us was to add more options to the DB update script: Customers might or might not have applied the BSA .st to their SmartDB
Basic Storage Areas

- An area for Data
- An area for Index‘es
- An area for LOB‘s
- A set of AI areas – just in case

- More fancy .st optimization may be applied later

```cpp
# b .
#
# "Schema Area":6,32;1 .
#
# "Data":42,128;8 .
# "Index":43,128;64 .
# "LOB":44,128;8 .
```
Agenda

- CCS
- Spec status
- Basic Storage Areas
- **Services and Managers**
  - Startup Manager
  - Service 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

```plaintext
/*
File: IManager
Purpose: This interface serves as a marker to indicate that a class is
        a common infrastructure service.

Syntax:
Description: Classes implementing this interface can be started and
             managed by the Startup Manager and are guaranteed to be
             available once the bootstrap process has completed.

Author(s): Simon L Prinsloo
Notes: The Startup Manager is the sole authoritative factory and
       repository of services implementing this interface.
*/

USING Ccs.Common.IService FROM PROPATH.

INTERFACE Ccs.Common.IManager INHERITS IService;

END INTERFACE.
```
Services in the SmartComponent Library

- Services have been the foundation of the framework since ever
- Foundation of customizability and extensibility
- Services (typically) identified by an Interface
- We did not require a blank constructor (when there was a specific factory)
- We did not have the requirement for an initialize() method
Services in the SmartComponent Library

- We are adding the IService interface through Interface inheritance to some of our services
- We have implemented the required CCS Managers either through wrappers around existing services or straight within the existing implementation
Agenda

- CCS
- Spec status
- Basic Storage Areas
- Services and Managers
  - Startup Manager
    - Service 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 remains started also completed launch
- Managers are accessed through the Startup Manager
- Services are accessed through the Service Manager
Ccs.Common.IStartupManager

/***************************************************************************/
File : IStartupManager
Purpose : The factory of all common components.
Syntax :
Description :
Author(s) : Simon L Prinsloo
Notes :
/***************************************************************************/

USING Ccs.Common.* FROM PROPATH.

INTERFACE Ccs.Common.IStartupManager INHERITS IManager:

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

METHOD PUBLIC IManger 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. Unable to customize.
Ccs.Common.Application class

CLASS Ccs.Common.Application FINAL:

/**************************************************************************
 Purpose: Provides access to the injected IStartupManager.
 Notes: I
**************************************************************************/
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 INITIAL '1.0.0'
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)

```plaintext
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
Startup Manager in the SmartComponent Library

- Launching of an environment has not been our strongest part
  - AppServer
  - Client or Batch

- Multiple elements:
  - Loading of a properties file
  - Loading of services or service factories from a services file
  - Setting of static configuration properties
  - Setting of custom log entry types (hard coded or through –param)
  - Some custom code, e.g. DB Aliases when required
New Startup Process around the Startup Manager

```
USING Consultingwerk.Util.* FROM PROPATH .

BLOCK-LEVEL ON ERROR UNDO, THROW .

{Consultingwerk/products.i}

DEFINE INPUT PARAMETER pcParameter AS CHARACTER NO-UNDO.

DEFINE VARIABLE oStartup AS IStartupManager NO-UNDO .

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

oStartup = NEW AppServerStartupManager (pcParameter) .
oStartup:initialize() .
```
Startup Manager Configuration

- Implemented JSON Config File
- Different sections
  - Configuration Values
  - CCS Managers
  - Service Definitions
  - Static properties
  - Custom Log Entry Types
  - Database Aliases
Config Files with Inheritance

- Basic SmartFramework (Consultingwerk)
  - AppServer for Web applications (Consultingwerk)
    - AppServer for specific application (customer)
      - AppServer with logging for a developer (customer)
- Sections can be inherited from base config file
- Sections can replace same section in base config file
"basedOn": "Consultingwerk/Framework/Server/startup_rest_appserver.json",
"inheritStaticProperties": true,
"staticProperties": {
    "Consultingwerk.Framework.FrameworkSettings:DebugMode": true,
    "Consultingwerk.OERA.DataAccess:LogFetchDataDetails": true
},
"inheritLoadServices": true,
"loadServices": [
    "Consultingwerk/SmartComponentsDemo/Web2/demo_services.xml",
    "Consultingwerk/OERA/TableStatistics/services_request_monitor.xml",
    "Consultingwerk/OERA/RestResource/services.xml"
],
"inheritCustomLogEntries": true,
"customLogEntries": [
    "ServiceInterface",
    "ServiceLoader",
    "ConfigurationProvider",
    "ServiceNameMappingService",
    "SmartWebHandlerRequest",
    "SmartHybridRealm",
    "SmartRepositoryService",
    "ContextDatasetStore",
    "Activation",
    "RestResource:4"
],
"inheritAliases": true,
"aliases": {
    "sports2000": ["dictdb", "appdb"],
    "icfd": "afdb"
}
{  
"basedOn": "",  
"inheritStaticProperties": false,  
"staticProperties": { },  
"applicationSettings": ".restapplicationsettings",  
"inheritLoadServices": false,  
"loadServices": [  
  "Consultingwerk/Framework/Server/rest_services.xml",  
  "Consultingwerk/SmartFramework/services_server.xml"  
],  
"inheritManagers": false,  
"managers": {  
},  
"inheritCustomLogEntries": false,  
"customLogEntries": [ ],  
"inheritAliases": false,  
"aliases": null,  
"sessionExport": null
}
New Style Config Files

- Services and application settings supported in two ways:
  - In place, in the JSON File
  - As a reference to previously used config files
    - Simplifies move to new startup process

- Resulting configuration written to (AppServer) logfile for reference and trouble-shooting of configuration related issues
Agenda

- CCS
- Spec status
- Basic Storage Areas
- Services and Managers
- Startup Manager
- **Service 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, …)
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.
Sample Service Manager implementation

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:PARAM), "=":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), 0).
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 DO:

  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
<?xml version="1.0"?>
<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.CustomService.IEuro2016ResultService</ServiceType>
    <InstanceName />
  </ttServicesRow>
</ttServices>
```
Accessing a Service

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

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

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

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

oMatchesService = CAST (Application:ServiceManager:getService
                        (GET-CLASS (IEuro2016ResultService)),
                        IEuro2016ResultService).

oResults = oMatchesService:GetTodaysMatchResults().
```
Service Manager in the SmartComponent Library

- **ServiceContainer** for (in session) services
  - Services typically implemented by Interface
  - We allowed for Services implemented by a Class

- **ServiceManager** for business services (exposed to the Service Interface) since ever
  - Typically Business Entities and Business Tasks
  - Business Services identified by a string identifier
  - May be a class name, a short class name or an alias
Service Manager in the SmartComponent Library

- Two different registries
  - Different use-case
  - Different type of keys: Progress.Lang.Class vs. CHARACTER
- ServiceContainer relies on external factory (or IServiceCreator registered instead of the Service)
- ServiceManager may use Alias-Service and launch a Business Entity
- Consultingwerk Service Container will not (always) error when requested Service type is not known; CCS requires to error when requested Service type is not known
Service Manager in the SmartComponent Library

- Implementation through a wrapper
- Combining ServiceContainer and ServiceManager
- Throwing error when requested service is not registered

- Service factories will invoke *initialize* method on new services, when Service implements CCS IService interface
- Service wrapper can also be used for non CCS services…
/**
 * Purpose: Returns a usable instance of the requested service
 * Notes: Throws an error when the service is not available
 * Calls into ServiceContainer:GetMandatoryService ()
 * @param poService The service type requested
 * @return The instance of the requested service
 */


DEFINE VARIABLE oInstance AS Progress.Lang.Object NO-UNDO .

IF poService:IsA ("Consultingwerk.OERA.IBusinessService":U) THEN
    RETURN ServiceManager:GetBusinessService (poService:TypeName) .

oInstance = FrameworkSettings:ServiceContainer:GetService (poService) .

IF NOT VALID-OBJECT (oInstance) THEN
    UNDO, THROW NEW ServiceNotRegisteredException (poService) .

RETURN oInstance .

END METHOD.
Agenda

- CCS
- Spec status
- Basic Storage Areas
- Services and Managers
- Startup Manager
- Service 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, count
  - 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 (e.g. RESTful clients)
Business Entities consumed by PSC products

- JSDO
- Kendo UI, Kendo UI Mobile
- Kendo UI Builder for OpenEdge
- ADM2, Dynamics (DataView SmartObject)
- 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 guessed result or a cached result
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, IGetDataTableRequest,
- 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,
 输出 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 PROPERTYTableRequests AS IGetDataTableRequest EXTENT NO-UNDO
   GET.
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

```ABL
oCustomerEntity = CAST (Application:ServiceManager:getService(GET-CLASS (IBusinessEntity),
               BusinessEntities:Customer:ToString()),
               IBusinessEntity).

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

- Our Business Entity already supported the complete CCSBE functionality
- All patterns for fetch supported, including count with caching and estimating
- Method names different: GetData() vs. fetchData()
- Implemented CCS Interface directly in the Consultingwerk.OERA.BusinessEntity class
- Typical customer’s Business Entities will inherit CCS compatibility without any code change
Demo

- Fetch data using ABL like Query String
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

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

```java
// SalesRep = "BBB" AND Country = "USA" AND NOT (City BEGINS "B" OR City BEGINS "C"
oRequest = NEW GetDataRequest (NEW GetDataTableRequest("eCustomer",
    QueryEntry:Array(
        NEW QueryPredicate("SalesRep", QueryOperatorEnum:Eq, "BBB"),
        NEW QueryPredicate (JoinEnum:And, "Country", QueryOperatorEnum:Eq, "USA"),
        NEW QueryGroup (JoinEnum:AndNot,
            QueryEntry:Array(NEW QueryPredicate("City", QueryOperatorEnum:Begins, "B"),
                NEW QueryPredicate(JoinEnum:Or, "City", QueryOperatorEnum:Begins, "C"))))).
```
Abstract Queries in Consultingwerk Business Entity

- Similar implementation was previously present
- Added CCS Interface to our primitive holder classes
- Added new Array holder classes
- Our own abstract queries based on Lists, not Arrays
  - CCS did not use Lists – as there are no set classes in the ABL
- Support for CCS abstract queries based on ability to convert our own list of Query Predicates to the CCS type
Demo

- Fetch data by abstract query definition
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 in Consultingwerk Business Entities

- CCS Named queries also similar to our own implementation

- Both CCS and SmartComponent Library named queries based on query name and list of arguments (values in primitive holders)

- We are typically generating specific query classes to make named queries strong-typed
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

```csharp
/*-----------------------------------------------
   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 (shou)
   @param poUpdateDataRequest The optional request object that allows to provide custom ir
   @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
Updating with Consultingwerk Business Entities

- Currently all our Business Entities implement IUpdatableBusinessEntity as update functionality is inherited from base
- Read-only Business Entities are possible (by missing the Before-Table and/or validation)
- We consider to implement non-updatable Business Entities by separating the functionality in the Base class into two classes
- Our own structure for ERROR-STRING is based on CHR(3)/CHR(4) delimited list (inherited from ADM2)
- Implemented switch to return CCS style JSON Array instead
Demo

- Update using CCS Interfaces and Consultingwerk Business Entity
Consultingwerk Business Entities go open-source

- We are preparing the release of the core CCS Business Entity (including Data Access) functionality under the MIT license
- Release on Github
- Estimated: Winter 2017/2018
Agenda

- CCS
- Spec status
- Basic Storage Areas
- Services and Managers
- Startup Manager
- Service Manager
- Business Entities

- Conclusion
Conclusions

- Implementation of CCS on top of OERA with ProDatasets is not hard! It's an evolution of OERA, not a revolution.
- SmartComponent Library has been very close to the specs already.
- Both on basic infrastructure and Business Entities.
- Implementation of the CCS specs allowed us to modernize a few key blocks in the framework (startup process, structured queries).
- Managed to implement CCS
  - Non breaking
  - In a way that our customers could leverage CCS without code changes.
Conclusions - Lessons learned

- CCS is a great forum for exchanging and shaping great ideas and experience
- Working in spec teams requires time; team 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 by far!

**Seeking more collaborators!**
Sample code

- Sample code on Github:
  - https://github.com/consultingwerk/CCS_Samples
Questions