SOA Developer’s Handbook

Contents

Welcome to Charlotte SOA Development

The City of Charlotte SOA Program was established in 2009 with the overall goal of improving City of Charlotte data and process collaboration capabilities. Prior to the creation of this program, most system integrations were created point to point as one-off solutions. This approach worked well for a long time, but as integration needs became greater and the business desire to access information from various sources increased, we saw a collective need to change our approach. In the spring of 2009, we formed a multi-KBU SOA Planning team to figure out how to address this. The net result of this analysis was the creation of an ongoing SOA Program, which is further defined by the program documents located here:

- SOA Program Charter
- Governance Framework
- SOA Standards
- SOA Reference
- SOA Project Submission Process
- SOA Project Submission Form

As a developer for the City of Charlotte, you will be responsible for following the guidelines setup by this program when developing any new integration or making any significant modification to existing integration. We have established an enterprise BizTalk platform to aid us in our efforts, so this is where your new SOA projects will ultimately need to reside. The intent of this guide is to help developers figure out how to do SOA Development as efficiently and effectively as possible.

Definition of Enterprise & Local Environments

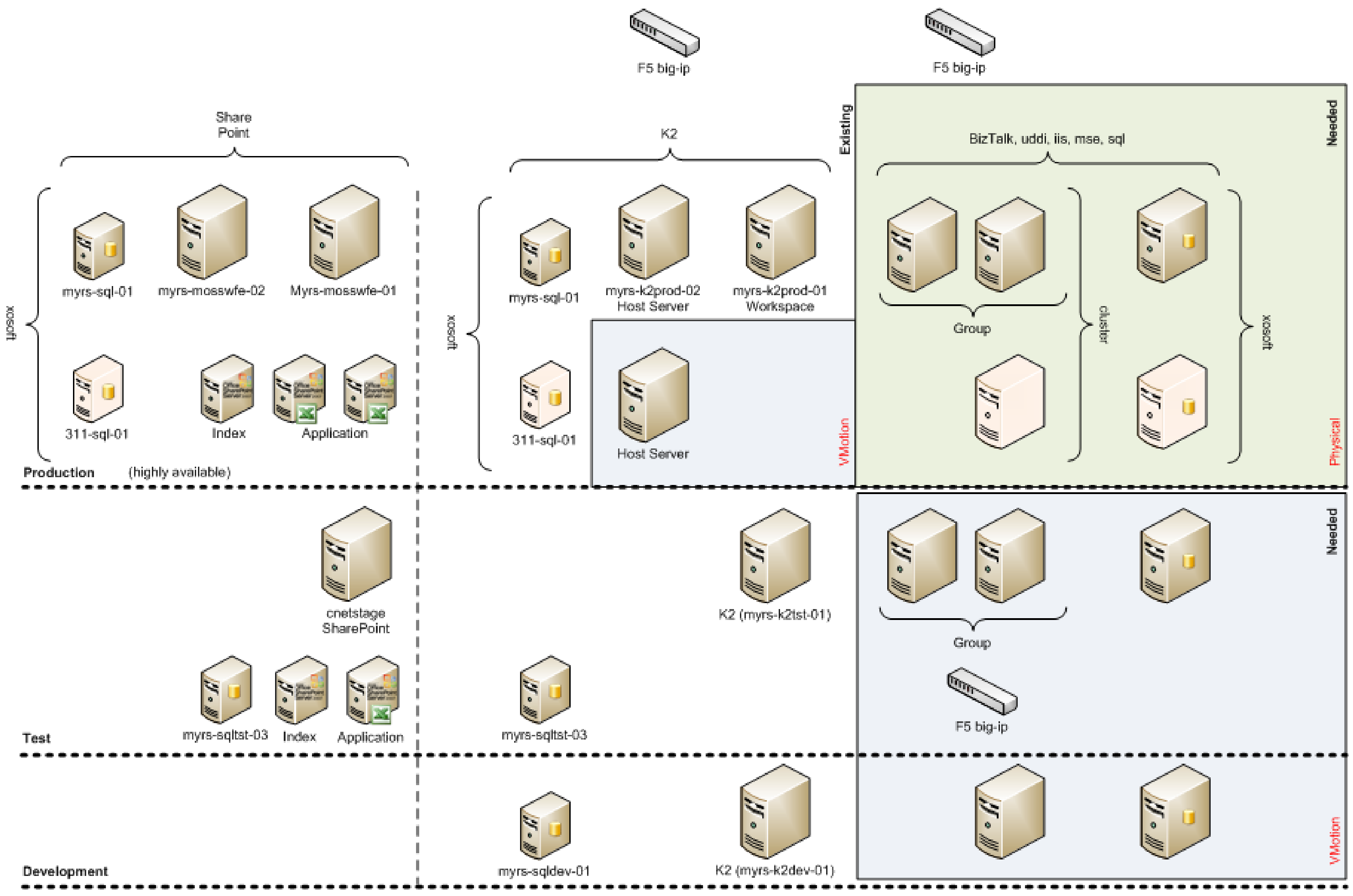
Multiple BizTalk environments are employed to enable SOA development. This section describes these environments and their purposes

Local Development Environment

Every SOA developer will need to have access to a locally managed BizTalk environment to do SOA project development. This could be an image local to your PC, a private image on a remote virtual machine, or an environment that is shared by developers in your group. It's really up to personal preference how this is approached. A distributable VMWare image along with instructions for setting it up has been created to make setting up the local development environment easier. This is optional, but if you wish to use it, instructions for doing so can be found here.

Enterprise Environment

The enterprise environment consists of three staging layers (development, test, and production) to aid in the deployment, testing, and operation of SOA projects. The server layout of these layers is illustrated in the diagram below and descriptions of the three layers are provided below that.



Enterprise Development Environment

This is the place where the code that the developer has written will be 1st deployed into the enterprise framework to ensure the bindings and project references work as they did during development.

Enterprise Testing Environment

Once deployment into the enterprise development environment has been successfully completed, the project can then be deployed into the testing environment. The purpose of the testing environment is to provide a place that mirrors the production environment where code can be tested one last time before going into production.

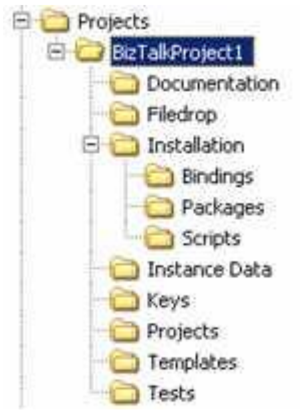
Enterprise Production Environment

This is the production operational environment. Projects should be deployed here only after all testing is successfully completed.

Local Configuration Guidance

Your local development environment is your own, but to make the development process easier, you should do a few things.

1. Create Hosts, Host Instances, and Adapters to mirror the enterprise environment: This will help to avoid issues with bindings that you've created not matching the configuration of the environment being imported to and causing import failures. The configuration should match what is documented here.
2. Store your BizTalk and associated files in a common directory structure. This article briefly discusses the common structure we'll use. The .vbs script presented is available here (you'll have to remove the .txt extension to execute it). The structure looks like this:



```
Projects
  BizTalkProject1
    Documentation
    Filedrop
    Installation
      Bindings
      Packages
      Scripts
    Instance Data
    Keys
    Projects
    Templates
    Tests
```

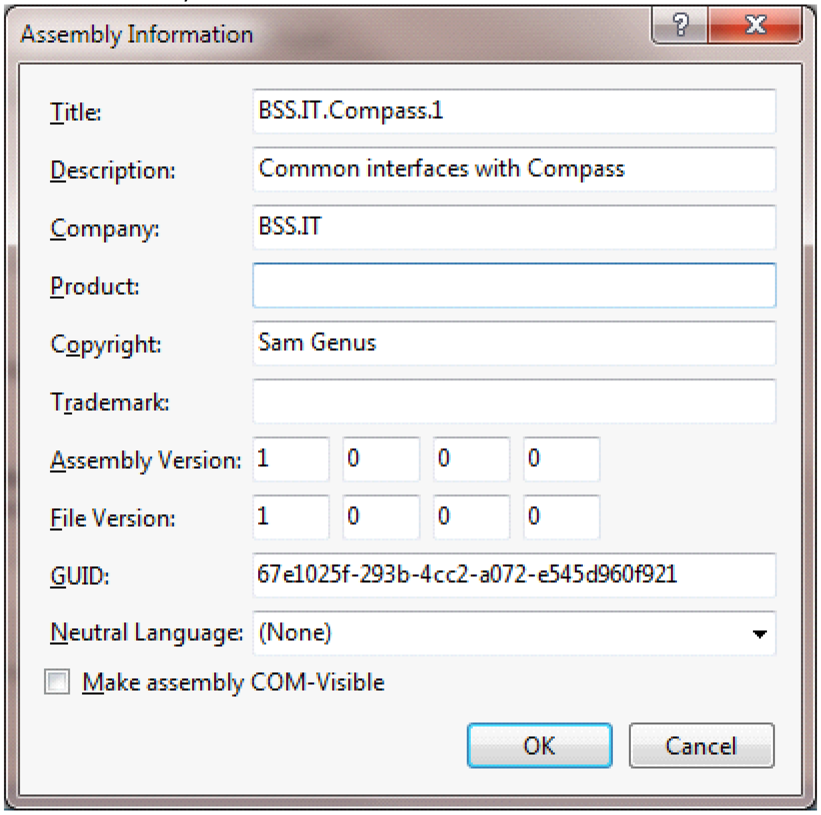
3. Installed enterprise shared artifacts as described in the Shared Enterprise Artifacts section below.
4. Install Microsoft Services BizTalk Documenter for documenting projects as part of the PTO submission process.

Project Development

Development Standards

- Naming conventions
  - Application Name: *<Department>_<Subject>_<BizTalk App Name>*
    - Department List: BSS, CMPD, CFD, CATS, CDOT, CMU, ENG, SWS, COC (enterprise)
    - Master Subject List:
      - Citizens (CI)
      - Public Interests (PI): traffic, the environment, art, etc
      - Regulations & Privileges (RP): permits, codes, transit passes, zones, etc
      - City Associates (CA): employees, contractors, elected officials, etc
      - Events (EV): service requests, complaints, crime, accidents, consumption, etc
      - Work (WK): work orders, projects, investigations, arrests, etc
      - Assets (A): buildings, evidence, roads, pipes, etc
      - Funds (F): budget, general ledger, bus fares, parking fees, payroll, etc
    - Example: *CMU_WK_WorkOrders_NewWorkOrder, COC_CA_Employees_NewEmployee*
  - Schemas

        - Root node: name should never be left as "root". Should always be renamed with a meaningful name that represents the type of doc described by the schema.
            - Example: *EmpBasicData*
        - Target namespace: $\texttt{http://charlottenc.gov/}\underbrace{\texttt{[Prop]Schema/<subject>/<AppName>/v1}}$
        - File name: <this with "_"'s>_<Root node>
            - Example(namespace): http://charlottenc.gov/Schema/Employees/NewEmployee/v1
            - Example(file name): *Schema_Employees_NewEmployee_v1_EmpBasicData*
        - Element naming: if an element reflects a thing (an object, a class or a table name), use UpperCamelCase; if it is a property, a reference, etc, use lowerCamelCase
            - Upper example: <PurchaseOrder>
            - Lower example: <dateTaken>1997-05-04</dateTaken>
    - Maps: <SourceSchema[meaningful representation]>2<DestSchema[meaningful representation]>
        - Example: *PeopleSoftData2EmpBasicData*
    - Orchestrations: <BizTalk App Name>_<meaningful name>
        - Example: *NewEmployee_ADNameLookup*
    - Receive / Send Ports, Groups, and Locations: <BizTalk App Name>_<meaningful name>_<transport type [FF, XML, SQL, etc]>
        - Example: *NewEmployee_PeopleSoftImport_FF*
- Application Properties
    - Application Tab
        - Target Framework should be .NET Framework 3.5
        - Basic information about the application should be stored in the Assembly Info (Assembly Information button) as follows:



Title:	BSS.IT.Compass.1
Description:	Common interfaces with Compass
Company:	BSS.IT
Product:	
Copyright:	Sam Genus
Trademark:	
Assembly Version:	1 0 0 0
File Version:	1 0 0 0
GUID:	67e1025f-293b-4cc2-a072-e545d960f921
Neutral Language:	(None)

☐ Make assembly COM-Visible

            -
            - Title should match the Application Name on the deployment tab

            - Even though description field only displays 1 line, it can contain a large amount of information (but no carriage returns)
            - Company should contain the authoring department name
            - Copyright should contain the technical contact name
    - Signing Tab
        - We will use 1 common .snk file for the organization
            - BSS will be the keeper & will distribute
            - If your project has sensitive data, you may create your own certificate
    - Deployment Tab
        - Configuration should always be changed to "Release"
        - Configuration Database & Server: should always point to developer's local environment
        - Application Name should never be left blank (if it is, artifacts will be written to default BizTalk application)
    - BizTalk Artifacts
        - Ports
            - Send ports should always be put into Send Port groups for future manageability
        - Schemas
            - Attribute vs Element
                - If you need a field to be nullable or have children nodes use elements
                - Otherwise, use attributes
            - Property Schemas
                - If a schema element may be needed as a searchable element at some point, it should be promoted to a property schema
        - Maps
            - Scripting functoids should only be used where built-in functoids are not available
        - Orchestrations
            - The first shape in every orchestration should be a scope shape
            - You should never have anything other than a receive shape as the start of a parallel action
            - Port binding: "Specify Now" port binding should never be used

Additional Development Guidance

Many things should be considered when developing a new BizTalk project to help make artifacts reusable and generally to advance on the goal of improving our data collaboration capabilities. To aid in this goal, general SOA design principles should be considered as they are discussed in our SOA Reference document.

Like many technologies, using BizTalk isn't necessarily complicated, but figuring out how to do something the first time can be time consuming. For this reason, it will be very important for us to share our discoveries so others in the organization can take advantage of them. We've created a Developer Forum for this purpose. Please feel free to use it as a starting point for figuring things out, but please also be sure to post your own findings back to it.

SOA Project Submission Form / Design Document

At some point during the project design phase, the SOA project idea will need to be brought to the SOA Center of Excellence for joint review. This could be done any time prior to actual development using the SOA Project Submission Form. If a lot of design guidance is needed, it could be brought early with not much detail other than a concept description. If little design guidance is needed, it could be brought later in the design with a more complete design document. It really depends on the individual developer's comfort level as to when this happens. An agreed upon design document should be completed prior to actual development to avoid the potential for having to do any re-work.

Deployment

Process Overview

		Who			**Where**			
Process Step		Developer	SOA CoE	BSS/IT	Local Dev Env	Ent Dev Env	Ent Test Env	Prod Env
1	Integration Concept	x						
2	SOA Project Submission Form	x						
3	Joint review	x	x					
4	Project development	x			x			
5	Test of deployment	x				x		
6	Submit PTO	x						
7	Deploy project for testing			x			x	
8	Test project	x					x	
9	Change Control	Primary		Secondary	*Note: developer & BSS/IT should coordinate to figure out what makes the most sense here*			
10	Deploy project to production			x				x

Local Development Environment

Development, including testing of bindings, should be done in the developer's local development environment. Once the project is ready for deployment into the enterprise environment, do the following:

- Setup the binding files
  - Export a separate binding file and name it Dev.xml
  - Make 2 copies to Test.xml and Prod.xml
  - Modify the binding file contents to reflect appropriate bindings for each enterprise layer
- Add the binding files as resources into the application
  - Go to Resources → Add
  - Add each binding file and match the "Target Environment" name at the bottom to the environment name (ex: resource file name = dev.xml, target environment = dev)
- Export an MSI file for the application, but on the "Select Resources" screen, *do not* include the bindings (when it is imported, the appropriate binding file will be applied from the attached resources)

Enterprise Development Environment

Once the .msi file has been created on the local development environment, developers should copy this to the enterprise development environment and test the deployment there. Since this environment will be shared across the enterprise, developers will need to schedule working time on the server to do this.

- Block time on the Enterprise Developer Environment Calendar
  - You may request assistance from a BSS deployment resource during this time as well
    - If requesting assistance, the developer will need to complete the Production Turn Over form to give the BSS resource information needed to assist
    - Call the help desk to setup a ticket for BSS/IT assistance
    - Request assistance by navigating to the calendar item details, clicking "Export Event", and then inviting whatever resource was assigned to the help desk ticket
- During the scheduled time, copy the .msi (and any other resource files) to c:\code\[username] on myrs-bizdev-01, and import it into BizTalk

Enterprise Test Environment

After successfully testing deployment in the enterprise development environment, developers should submit a Production Turn Over request to have a BizTalk application installed into the Enterprise Test Environment.

- Document application with BizTalk Documenter. Be sure to select your application as the document scope and set the output to Word 2003Xml.
- Complete a Production Turn Over submission and be sure to attach a zipped archive of the BizTalk documenter output, .msi file, and any other necessary deployment files
  - Any deployment dependencies such as the creation of a new database, new adapter, etc can be listed in the PTO
- BSS will deploy to test within 2 business days unless a later date is specified in the PTO
- Original developer is responsible for testing (accounts will be added to the BizTalk Server Operators group as described here). If for any reason (like testing failure), this code will not be promoted to production, it must be removed from test within 15 business days of installation to ensure test and production environments remain in synch
- If this code will be going to production, a change control for promotion must be submitted within 30 days of testing

Enterprise Production Environment

Once functional testing of the BizTalk application is complete, the standard change control process should be used to request promotion of the application to the production environment. Visibility to BizTalk and server logs will be made available through a viewer tool such as BizTalk MsgBoxViewer.

Note: the same PTO and associated files submitted for the Enterprise Test Environment will be used for the Enterprise Production Environment.

Shared Enterprise Artifacts

On occasion, project teams or the SOA CoE may identify an artifact as a shared enterprise artifact, such as an enterprise schema (ex: Addresses). To make these artifacts available for shared usage, they should be isolated into separate applications with the naming convention COC.<Artifact Type>.<Artifact Name (including version)>. After the shared artifacts are promoted to production, their .MSI files will be shared at \\cmgc-backup-02\DBAVirtualPC$\BizTalk\SharedArtifacts . Developers should load needed shared .MSI's into their local development environments as the need arises (be sure to install it in the GAC after it's imported into BizTalk). If a new version of a shared artifact is needed, the developer should create it (with CoE sign-off) and deploy it just as described above, but with a new version number. For consistency, each shared artifact project should contain only 1 shared artifact. As a reminder, a complete list of artifacts (not segregated by project), can be viewed in BizTalk under the "<All Artifacts>" project pseudo name.

Future Topics

- Need a process for undeployment. Will address this when the need arises.
- UDDI usage