BizTalk Code Review Checklist

Naming Standards Review

Standard	Result		Correction Details
	Pass	Fail	
Visual Studio.NET solution name follows convention of: [Company].[Dept].[Project]			
Visual Studio.NET project name follows convention of: [Company].[Dept].[Project].[Function]			
Schema name follows convention of: [RootNodeName]_[Format].xsd			
Property schema name follows convention of: [DescriptiveName]_PropSchema.xsd			
XSLT map name follows convention of: [Source Schema]_To_[Dest Schema].btm			
Orchestration name follows convention of: [Meaningful name with verb-noun pattern].odx			
Pipeline name follows convention of: Rcv_[Description].btp / Snd_[Description].btp			
Orchestration shape names match BizTalk Naming Standards document			
Receive port name follow convention of: [ApplicationName].Receive[Description]			
Receive location name follows convention of: [Receive port name].[Transport]			
Send port name follows convention of: [ApplicationName].Send[Description].[Transport]			

Schema Review

Standard	Result		Correction Details
	Pass	Fail	
Namespace choice consistent across schemas in project/name			
Nodes have appropriate data types selected			
Nodes have restrictions in place (e.g. field length, pattern matching)			
Nodes have proper maxOccurs and minOccurs values			
Node names are specific to function and clearly identify their contents			
Auto-generated schemas (via adapters) have descriptive file names and "types"			
Schemas are imported from other locations where appropriate to prevent duplication			
Schemas that import other schemas have a "root reference" explicitly set			
Clear reasons exist for the values promoted in the schema			
Schema elements are distinguished appropriately			
Schema successfully "validates" in Visual Studio.NET			
Multiple different instance files successfully validate against the schema			

Mapping Review

Standard	Result		Correction Details
	Pass	Fail	
Destination schema has ALL elements defined with either an inbound link, functoid, or value.			
Functoids are used correctly			
Scripting functoid has limited inline code or XSLT.			
Scripting functoid with inline code or XSLT is well commented			

Database functoids are not used			
Multiple “pages” are set up for complex maps			
Conversion between data types is done in functoids (where necessary)			
Map can be validated with no errors			
Multiple different input instance files successfully validate against the map			

Orchestration Review

Standard	Result		Correction Details
	Pass	Fail	
Each message and variable defined in the orchestration are used by the process			
Transactions are used appropriately			
All calls to external components are wrapped in an exception-handling Scope			
No Expression shape contains an excessive amount of code that could alternately be included in an external component			
The Parallel shape is used correctly			
The Listen shape is not used in place of transaction timeouts			
All Loops have clearly defined exit conditions			
Where possible, message transformations are done at the “edges” (i.e. port configurations)			
Calling one orchestration from another orchestration is done in a manner that supports upgrades			
Correlation is configured appropriately			
All messages are created in an efficient manner			
The message is not “opened” in unnecessary locations			
All variables are explicitly instantiated			
No port operations are named the default “Operation_1″			
Port Types are reused where possible			
All Request/Response ports exposed as a web service are equipped with a SOAP fault			

message.			
Orchestration has trace points inserted to enable debugging in later environments			
Orchestration design patterns are used wherever possible			

Business Rule Review

Standard	Result		Correction Details
	Pass	Fail	
Business rule output tested for all variations of input			
Conflict resolution scenarios are non-existent or limited			
Long-term fact retrievers used for static facts			
Business Rule vocabulary defined for complex rule sets			

Configuration Review

Standard	Result		Correction Details
	Pass	Fail	
Receive Port / Send Port tracking configurations appropriately set			
Maps are applied on the Receive Port where appropriate			
Send port retry interval set according to use case			
Maps are applied on Send Port where appropriate			
Send port does NOT have filter attached if connected to an orchestration			
Subscriptions exist for every message processed by the application			

Deployment Package Review

Standard	Result		Correction Details
	Pass	Fail	
“Destination Location” for each artifact uses “%BTAD_InstallDir%” token vs. hard coded file path			
All supporting artifacts (e.g. helper			

components, web services, configuration files) are added as Resources			
Binding file is NOT a resource if ports use transports with passwords			

Overall Solution Architecture Review

Standard	Result		Correction Details
	Pass	Fail	
Solution is organized in Visual Studio.NET and on disk in a standard fashion			
Passwords are never stored in clear text			
All references to explicit file paths are removed / minimized			
All two-way services INTO BizTalk produce a response (either expected acknowledgement or controlled exception message)			
Calls to request/response web services that take an exceptional amount of time to process are reengineered to use an “asynchronous callback” pattern			
Exceptions are logged to an agreed upon location			
Long-running processes have a way to inspect progress to date			
Solution has been successfully tested with REAL data from source systems			
Solution has been successfully tested while running under user accounts with permissions identical to the production environment			
Messages are validated against their schema per use case requirements			
Processes are designed to be loosely coupled and promote reuse where possible			