

WEB04 - TLC Commerce Server 2007 Architecture Deep Dive

David Messner

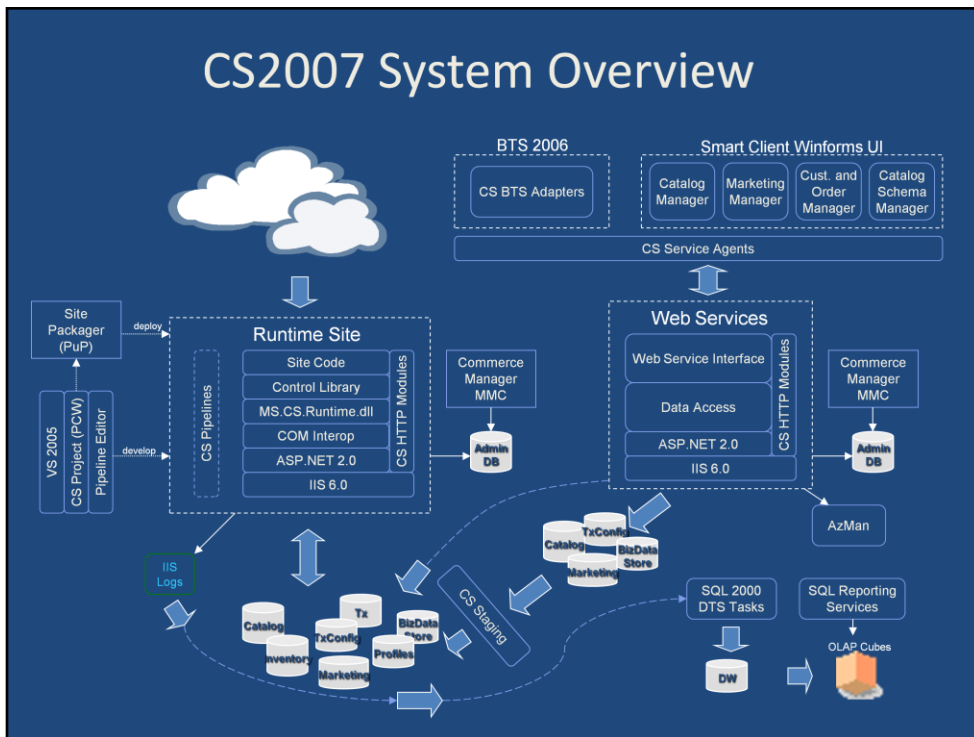
Abstract:

This talk will provide an overview of the inner-workings of Commerce Server 2007 and how the various components fit together. Includes details of the overall architecture with deep-dives on Catalog/Inventory, Profiles, Marketing, Orders, Data Warehouse, Adapters, and Staging services.

These are an adaptation of the slides from the CS2007 Architecture Overview TLC presentation given at Microsoft TechEd 2007 in Orlando, FL.

Recording of the previous architecture WebCast available here:

[https://msevents.microsoft.com/CUI/WebCastEventDetails.aspx?EventID=1032303895
&EventCategory=5&culture=en-US&CountryCode=US](https://msevents.microsoft.com/CUI/WebCastEventDetails.aspx?EventID=1032303895&EventCategory=5&culture=en-US&CountryCode=US)



One of the common mistakes that businesses make when developing their e-commerce sites is to invest a lot in the development of their e-commerce front-end with little consideration to the ongoing operation and management of the site. This is where Commerce Server can really help a business reduce the TCO of their e-commerce solution. This slide shows many of the components of a full Commerce Server deployment and how they fit together in the site's entire production lifecycle.

The typical first step in the lifecycle of the site is the development of the runtime application. The site developers can choose to start with the Commerce Server 2007 Starter Site (a separate download from the main product install), or to start with a basic site template by using the Commerce Server Project Creation Wizard in Visual Studio 2005. Once the site is developed, the Site Packager tool, or PuP, can be used to deploy the site content and associated schemas onto the production servers or into a staged environment.

A Commerce Server deployment is defined by the set of servers that reference a specific Commerce Server administration database. As part of the installation of the product, you specify an administration database to connect to or to create. The Commerce Server Manager MMC snap-in can then be used to manage the connection strings and configuration properties for the various Commerce Server resources.

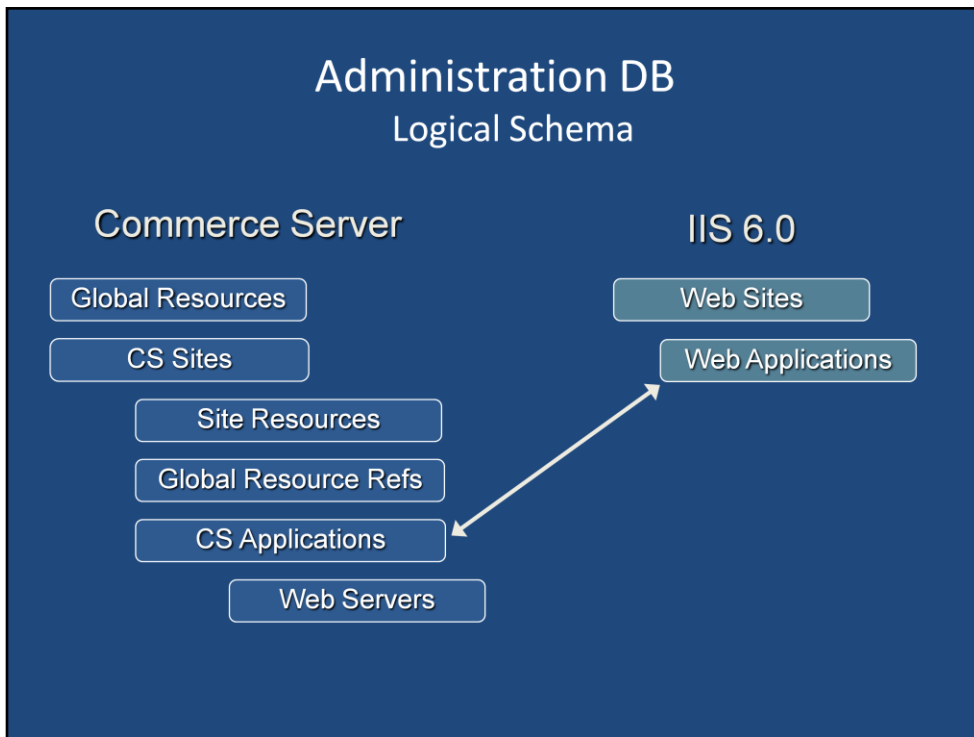
Commerce Server also includes a set of applications that the site managers can use to create and manage the business data of the site such as the product catalog data and the marketing campaigns that are currently in effect. These applications include the Catalog Manager, Marketing Manager, the Customer and Order Manager, and the Catalog and Inventory Schema Manager. Each of these communicates with the CS Web services over HTTP and SOAP, allowing for simple, secure, and interoperable deployments both within the firewall and outside of it.

The business data is typically updated within a test or staged environment, and that's depicted here by a set of servers using a separate administration database from the production deployment. Once the data and any corresponding site code updates have been tested and validated, the Commerce Server staging service can be used to push the updates out into production in either an on-demand or scheduled fashion.

As the site operates, traffic information is captured in the IIS log files and transactions and other data are collected within the site databases. Commerce Server includes powerful data warehousing and analytics capabilities for aggregating and reporting on this data. SQL Server DTS tasks are used to import data from the log files and transform data from the runtime databases into the data warehouse system. OLAP cubes are then generated which can then be reported on from SQL Reporting Services reports.

And of course the Biztalk adapters can be used to periodically ship new orders out into external systems for fulfillment or even to accept baskets and orders from other sources into the system. Catalog, inventory and profiles data may also be imported and exported through the adapters.

Not shown on this diagram are the CS2007 MOM pack and the Commerce Server Health monitoring service. The health monitoring service can be used to periodically ping the Commerce Server Web services and ensure that they are configured properly and are responding to requests in a timely fashion. It will raise events that are picked up and handled by MOM in case of a problem.



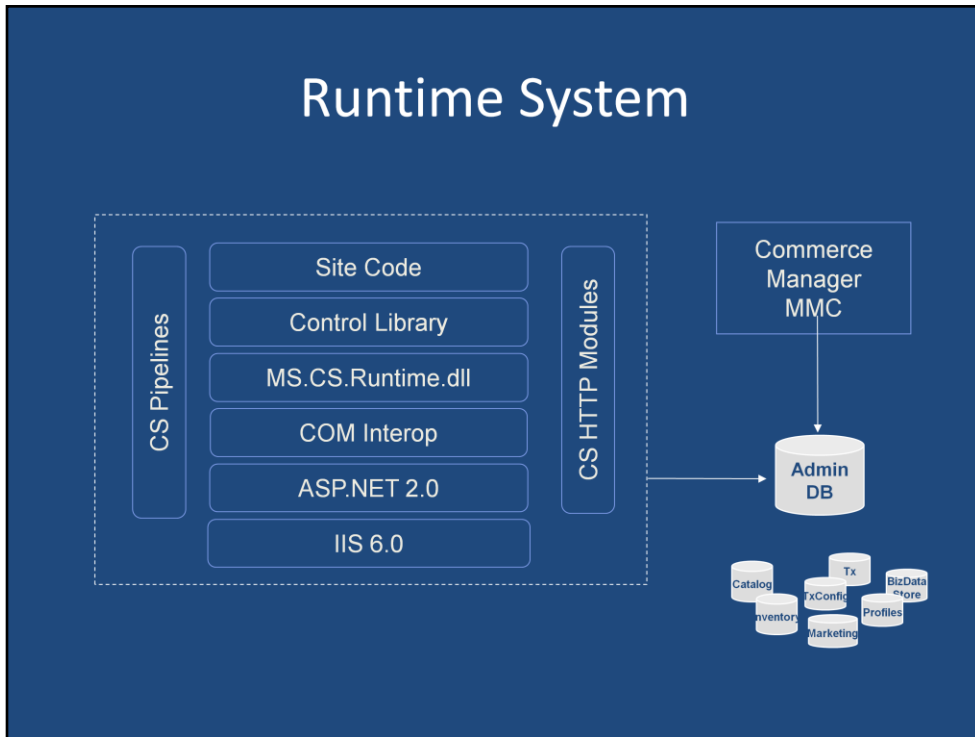
This slide depicts the logical schema of the Commerce Server administration database. A Commerce Server Site consists of a set of resources which may be global or site level resources. Global resources can be shared across sites whereas site resources are local to the site. Global resources include the profiles and direct mail resources and the now deprecated Commerce Server Authentication resource. Site resources include the catalog, inventory, transactions, transactions config, and marketing resources.

Resources are typically associated with a database. So the configuration for a resource may include one or more connection strings and a set of configuration properties. The unpack action of the site packager tool is used to create sites and resources. As a best practice, each resource should be unpackaged to a separate SQL database.

Commerce Server applications correspond to IIS 6.0 Web applications. A typical Commerce Server site will include one main runtime application and the four Commerce Server Web service applications. However, more applications may be added to a site, and each application can share the site resources for that site. Custom resources can also be created.

Notice that while there is a mapping between Commerce Server applications and IIS Web applications, there is no correlation or mapping between the concept of a Commerce Server Site and an IIS 6.0 Web site.

Runtime System



Commerce Server sites are ASP.NET 2.0 Web applications. The main classes and types that are used in the development of a runtime site are contained within the Microsoft.CommerceServer.Runtime assembly. This assembly includes a set of HTTP Modules and configuration section handlers that provide the initialization and configuration services for the various subsystems of Commerce Server. These handle tasks such as creating and configuring the runtime site caches, pipelines, and the runtime contexts for the various subsystems.

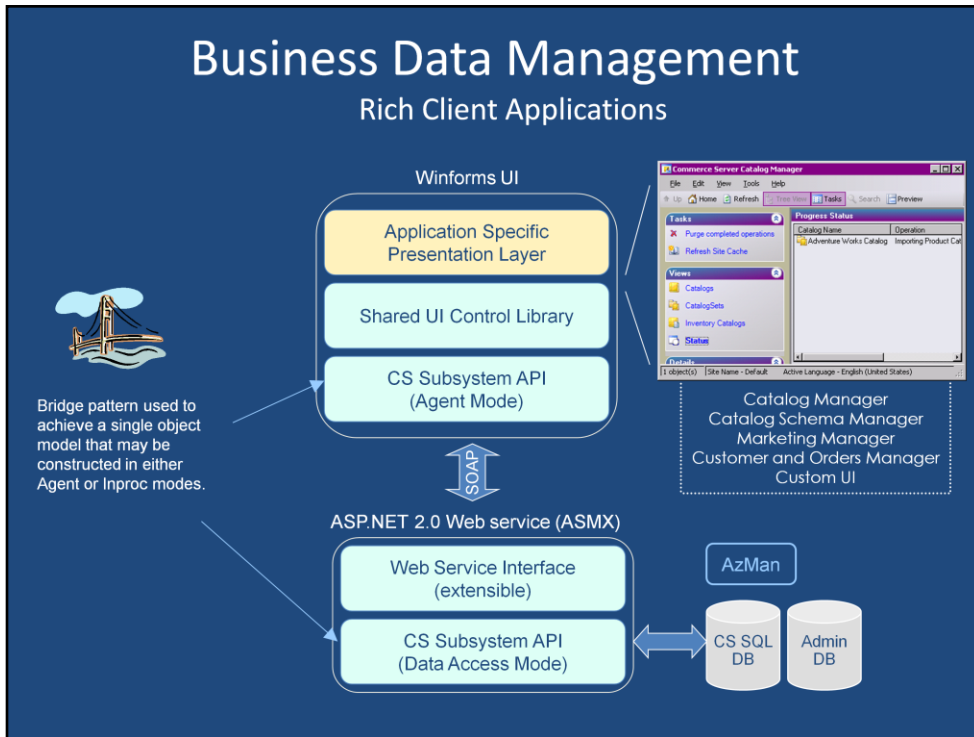
As I mentioned earlier, we've made a big investment in moving more of Commerce Server to a fully managed stack, however there are still some parts that utilize COM interop. This is almost entirely hidden from the site developer, yet some of the more advanced extensibility scenarios do require you to interoperate at the COM level. The marketing runtime and orders pipelines, for example, still utilize COM internally. So extending these systems does require you to expose code via COM. However you're still free to implement your components in C# or Visual Basic .NET, and in fact we've improved the interop performance considerably in both of these cases by allowing managed pipeline components to be pooled in COM+ whereas that was not possible in Commerce Server 2002.

Notice that the site architecture utilizes an In-process model. This has the benefit of very high performance, particularly if sticky load balancing can be utilized to maximize the caching of user profiles on the Web server.

We often get the question whether Commerce Server can run as an application server on a separate tier from the Web servers. The answer is that this is not possible out of the box since several of the runtime APIs are not remotable. With some work you could achieve this by building your own remoting layer using Windows Communication Foundation or one of the other .NET remoting methodologies.

Business Data Management

Rich Client Applications



As we saw on the system overview diagram, each of the business management applications included with CS communicates with the back end through SOAP Web service calls. Each of the subsystems includes an API layer that we call the Agent API's. These Agent mode API's expose the functionality of the Web service through a hierarchical object model rather than the flat Web service interface, making them easier to program against. They also handle services including authentication and exception handling. When the client connects to the Web service using the Agent API's, the ServiceAgent class negotiates an authentication method with the server. All of the standard IIS authentication mechanisms are supported with NTLM authentication being the default. Basic authentication is supported, however by default the agents will only authenticate using Basic authentication over a secure SSL channel. You can override this setting either programmatically or through the application configuration files if you want to test with Basic authentication but without requiring SSL.

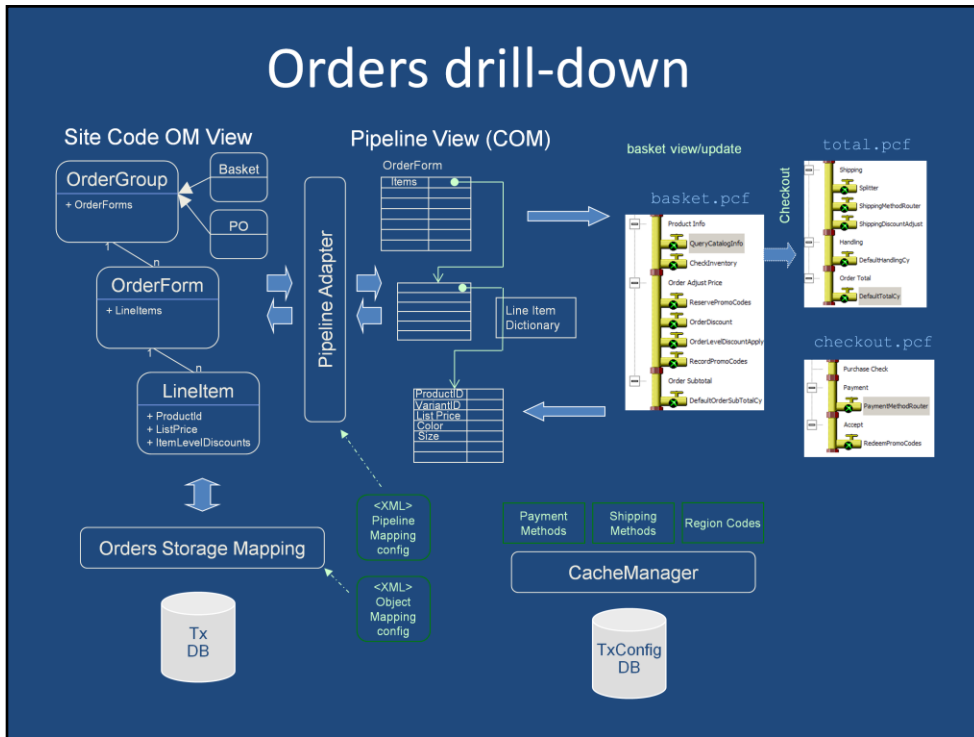
Once a user is authenticated by the Web service, task-based authorization is performed by using Windows Authorization Manager, also known as AzMan. This provides a great deal of flexibility for the business, since AzMan policies are easily customizable. At the lowest level, operations are defined corresponding to each of the capabilities of the Web service. These are rolled up into higher level tasks within the policy. Note that there is a binding in the applications to specific task names in the AzMan policies, so these task names should not be changed if you are using the out of box applications.

Typically when you invoke a Web service method and an error occurs on the back-end, a SoapException is returned to the client. But when you make calls using the Agent API's, there is a special exception serialization and propagation mechanism used that allows the server to serialize exception details and send them to the client where the typed exception is deserialized and rethrown. Note that for security reasons not all exception information is propagated to the client. Any exception that may disclose information to a hacker or which wouldn't be of interest to a business user is logged to the event log on the server and a more generic ServerFaultException is returned to the client. A SqlException is a good example of this.

These same Agent API's can also be initialized to run in-process, which we call the local or data access mode of operation. This duality of implementations is achieved by internally implementing the bridge design pattern where an implementation object is selected at runtime based on the construction overloads that are used. This is supported in each of the data management APIs except for profiles which already has a set of API's that can be used in-process for the same use cases.

As a final point on this slide, if deep customization of the Business user apps is required, the sources are available in the Commerce Server Partner SDK.

Orders drill-down



At the heart of the orders system for CS2007 is a completely rearchitected storage engine. The DBStorage object from previous releases has been replaced with an object-relational mapping engine. This engine consumes XML configuration files that define the mapping between the orders system entities and the database schema. You can inherit from and extend each of the orders objects. There are a number of benefits that are realized from this including a much more discoverable runtime object model that uses strongly typed properties instead of property bags, and more control over storage. These entities also know how to serialize and deserialize themselves into and out of XML. Weakly typed property collections are still available on each of the entities in order to support quick and dirty customization scenarios.

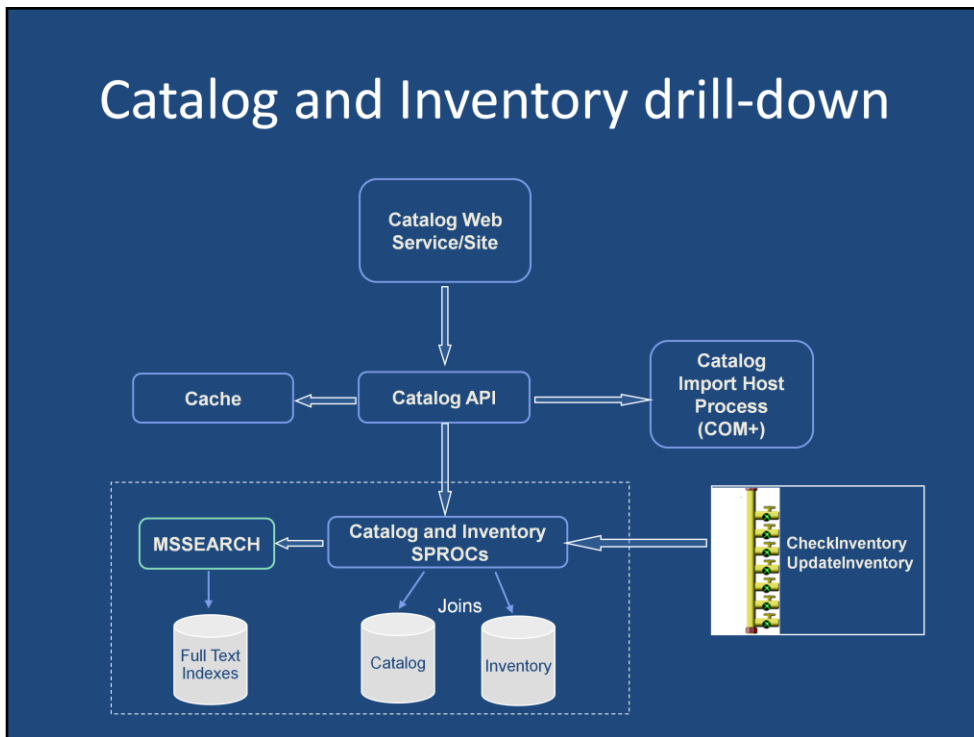
For basket and order processing, the pipeline architecture from previous releases is used, so your investments in pipelines continue to be leveraged. A pipeline is nothing more than a defined sequence of stages and components for pluggable business logic. For example, there are components provided for retrieving extended properties of line items from the catalog, for verifying promotion codes and for calculating discounts and shipping. Third party payment components are available from companies such as Cybersource.

Pipeline components are exposed as COM components but may be implemented in .NET. Pipelines operate on the dictionary and simplelist COM objects, and this is where the pipeline adapter comes into place. Whenever a pipeline is executed through the order system, the entity properties are marshalled into the corresponding COM objects and similarly after the pipeline finishes executing they are marshalled back to the strong-typed object model. The mapping performed by this layer is customizable and extensible by modifying the Pipeline Mapping XML configuration file.

When a checkout transaction occurs, the execution of the checkout pipeline and the persistence of the order to the database should occur together in the same DTC transaction. You can accomplish this by adding the transacted page attribute to the checkout.aspx page or by defining a System.Transactions TransactionScope within your code.

The final component to notice on this diagram is the CacheManager. CacheManager reads the caches section from Web config and constructs a collection of CommerceCache objects based on it. The orders system utilizes these caches for configuration information include payment methods, shipping methods, and region codes to avoid I/O that would otherwise be involved in accessing this slowly changing data. The caches can be configured to refresh periodically and cache refreshes can be initiated through the Web services or business applications and also triggered by staging events.

Catalog and Inventory drill-down



Architecturally, the Product Catalog and Inventory system is one of the simplest subsystems – it closely follows the Agent/Service pattern from the earlier slide; in fact it is the purest implementation of that since there is no distinction between runtime and datamanagement APIs in the catalog as there is in the orders and marketing systems. Most of the complexity is actually in the data layer where all of the heavy lifting is performed. Because of the dynamic nature of the catalog schema, the stored procedures in this layer do a lot of dynamic SQL query generation and execution and the optional presence of the inventory resource adds even more to that complexity.

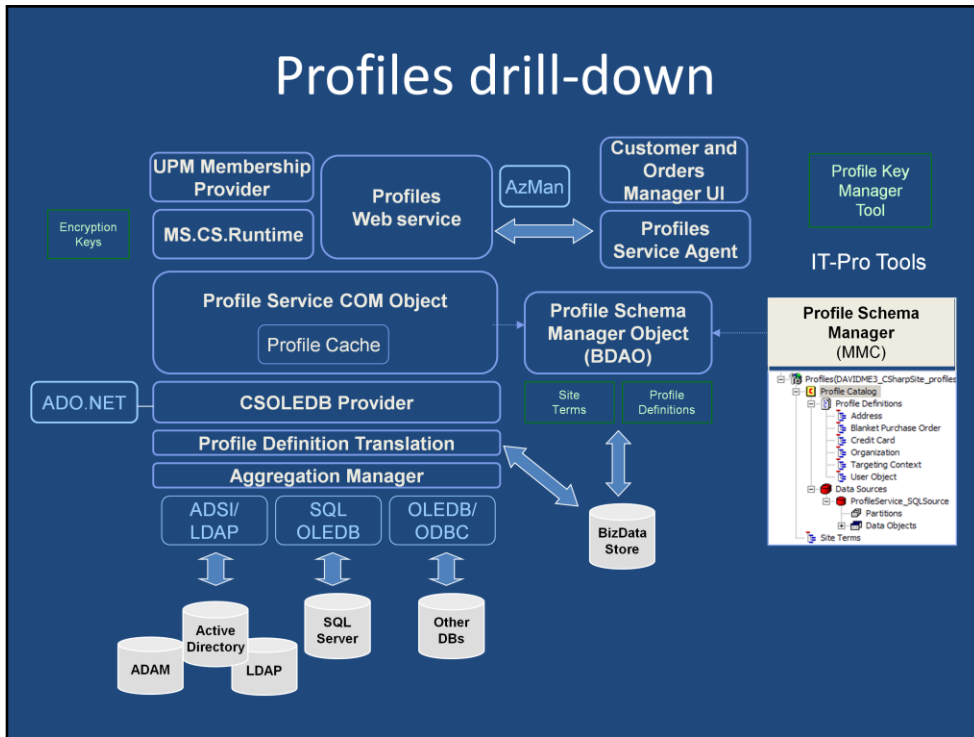
While Catalog and Inventory are separate Commerce Server resources, they are serviced by a single object model and Web service. The inventory resource cannot be used without a Catalog resource but Catalogs can exist without inventory. When the inventory resource is present, Catalog data is joined with inventory information at the data layer so that the number of round trips to the database is minimized.

You'll notice here that Catalog and Inventory can reside in separate databases. We even did some testing with Catalog and Inventory hosted on different SQL servers using the linked servers feature of SQL server, but we encountered some performance limitations with that, so it's not supported. You'll realize the best performance if Catalog and Inventory are in the same database, and this is the default when you unpack a new site.

Long running processes invoked through the catalog system such as imports and virtual catalog rebuilds are executed in a separate process hosted by COM+ and inherit the security context of the caller. This is done to decouple the lifetime of the catalog operations from the Web site or Web service process, which may be recycled at any time.

Inventory integration with the orders system is achieved through pipelines. The basket pipeline includes a CheckInventory component that will determine the inventory condition and stock levels for each product in the basket. If the product is out of stock, the component will issue a warning result code which will prevent a checkout transaction from completing when an item is out of stock.. The checkout pipeline includes the UpdateInventory component which is used to decrement the available inventory for the products being checked out.

Profiles drill-down



The Commerce Server profiling system offers capabilities that go well beyond what ASP.NET provides. Perhaps most notably it's not limited to the notion of a single profile representing users – Commerce Server also includes out of the box profiles for other entities including Addresses, Organizations, CreditCards, and the Targeting information for use with the Marketing System. It also includes a powerful aggregation layer that allows for a consolidated logical view of a profile that may actually be split across multiple data stores including Active Directory and SQL and an internal profile cache for maximum performance. It's also been tested to a much higher scale and has been deployed with up to 60 million users.

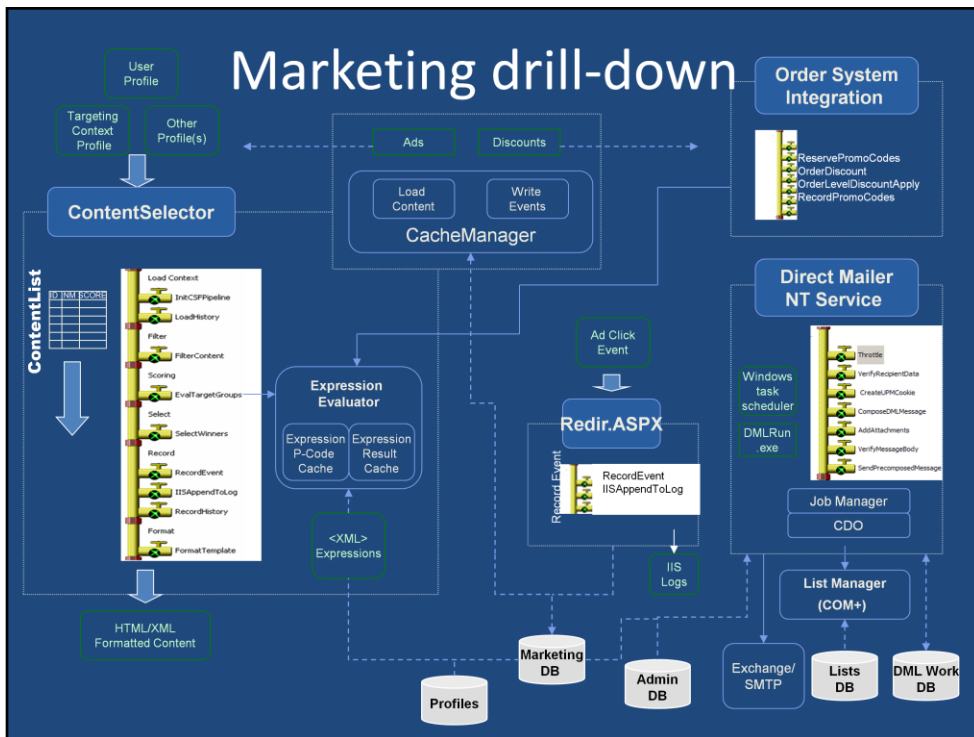
The Commerce Server OLEDB provider lies at the heart of the profiles capabilities. The provider loads the catalog of profile definitions from the Biz Data Store resource and maintains the mapping between the logical profile definitions and the physical data sources. It uses the services of the Aggregation Manager layer and the individual data providers for the physical profile data access and updates. CRUD operations on profile instances are done through the Profile Service COM object or the ProfileContext .NET object that is layered on top of it and which is part of the MS.CS.Runtime assembly. For multiple instance queries or bulk updates, the CSOLEDB provider can be used directly through ADO.NET. In either case, the logical schema mapping isolates the developer from the details of the physical data sources and from future changes that might be made to them.

Schema management is performed via the Profile Schema Manager, also known as the BDAO, or Business Data Admin Object. This component permits import and export of profile and data source definitions as XML. The Profile Schema Manager MMC component and the profiles PuP object use the services of this component for schema management.

New for CS2007 are these layers at the top, the UPM Membership provider and the Profiles Web service. The UPM Membership provider is a full-fledged implementation of an ASP.NET 2.0 membership provider and permits usage of the authentication and membership controls included with ASP.NET. On the topic of authentication, I'd also mention that the Commerce Server Authentication components from CS2002 including AuthManager and AuthFilter are still available in the product but are deprecated in this release. Our guidance is to use ASP.NET's authentication capabilities for new sites, the capabilities are greatly matured in the 2.0 release and have closed the capability gap between ASP.NET authentication and Commerce Server authentication.

The Profiles Web service aggregates the capabilities of the CSOLEDB provider, the Profile Service, and the Profile Schema Manager components through an XML-centric interface. The Profiles Service agent API, in turn, provides helper methods that ease the creation and consumption of these XML payloads.

The profiles system has support for encryption including both one-way hashing for secure storage of passwords and asymmetric encryption for secure storage of sensitive data such as credit cards. The Profile Key manager tool is a new command line tool that can be used to generate encryption keys and store them securely in the registry instead of putting them directly in the Web.config file. It also supports rolling the encryption keys and re-encrypting profile data with a new key.



The Marketing System runtime in CS2007 has incremental capability enhancements over the previous versions. Core capabilities continue to include a targeted advertising engine, a rich discounting system, and a DirectMailer service for personalized e-mail campaigns.

The most notable architecture changes for the Marketing System are the addition of the promotion code capabilities that were first introduced in Commerce Server 2002 Feature Pack 1, an all-new ListManager that is no longer a Windows service but is now implemented as a COM+ Enterprise Services component and which features tighter integration with targeting expressions, usage of the Windows task scheduler for scheduling of direct mail jobs instead of the SQL server agent, and an all-new Web service and Data Management API layer that was sorely missing from CS2002.

The Marketing system utilizes the pipeline architecture, which allows for extensibility and customization of the business logic. Personalized advertising and discount display is facilitated through the ContentSelector object which takes as input a set of profiles representing the user and other context properties of the request, executes a ContentSelection pipeline, and uses XML targeting expressions to score and rank the content. Targeting expressions are what affords the system great flexibility. The expression evaluator component that executes these expressions uses intermediate result caches and an optimized p-code representation to speed evaluation.

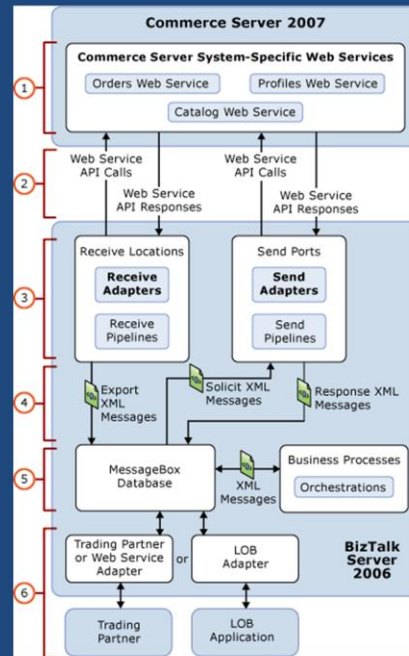
Marketing system integration with orders is also through pipelines. The OrderDiscount pipeline component, which is part of the basket pipeline, contains the core discount evaluation algorithms and once again uses targeting expressions and the expression evaluator to yield a great deal of flexibility.

The promotion code system is also implemented as a set of pipeline components. Promotion codes can be public, meaning available for anyone's usage, private, meaning they are sent to a specific user and their redemption may be tracked back to that user, and protected, which is the same as private except that the user's identity is also verified at the time of redemption so that only the targeted user can receive the discount. Promotion codes can have an associated usage limit or may allow unlimited usage. Promotion code reservations are supported for limited use codes, which means that if a user expresses an intent to use a code by entering it into the system, it will be reserved for them until some time limit expires in which case the usage of the code is returned back to the available pool.

Campaign events including ad display, ad clicks, ad image downloads, and discount redemption events are logged to the IIS log file together with the page request, allowing them to be imported to the data warehouse to support the generation of campaign reports.

Biztalk Adapter drill-down

- Adapters communicate with CS exclusively via Web services
- Receive adapters and Send adapters
- Additional message processing can be done in send/receive pipelines
- XML Messages (e.g. export, query, import, submit, update)
- Orchestrations, data transforms
- External system: LOB, trading partners, other CS deployment



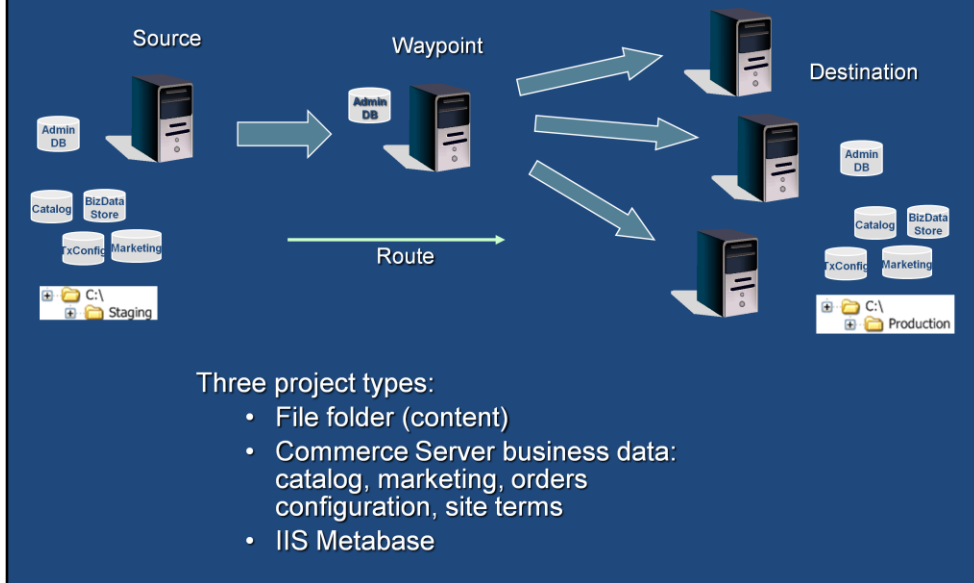
Commerce Server provides Biztalk adapters for orders, catalog, inventory, and profiles. Both send and receive adapters are provided for each subsystem. The message types supported include export, import, query, submit, and update. Send adapters use the Solicit/Response pattern where a solicit message such as a query or export message is sent to Commerce Server through the adapter and the resulting XML response body is returned back and put onto the Biztalk MessageBus. In the case of a submit, import, or update message, this response simply reports the success or failure of the operation. Receive adapters are typically used to export newly captured or changed data from CS, so for example exporting all of the new orders that have been collected through the site into Biztalk for final processing or fulfillment.

One of the key architectural points here is that the adapters use the services of Commerce Server exclusively through the Web service interfaces. This allows you to effectively decouple your Biztalk and Commerce Server systems. One of the challenges of this architecture was providing for distributed transactions that can span both the Commerce Server and Biztalk nodes. This is accomplished by passing a token representing the DTC transaction between the adapter and the Web service. The transaction token is passed through an HTTP cookie as part of the Web service calls.

By using the adapters to connect Commerce Server with external systems, you get all of the benefits of Biztalk server such as orchestrations, schema mapping, and Biztalk pipelines for additional message processing and transformations.

There's an excellent technical deep dive screencast available on the CommerceServer Technet site and which was produced by the product team. I've only scratched the surface here, I hope you will check out the screencast to learn more.

Commerce Server Staging

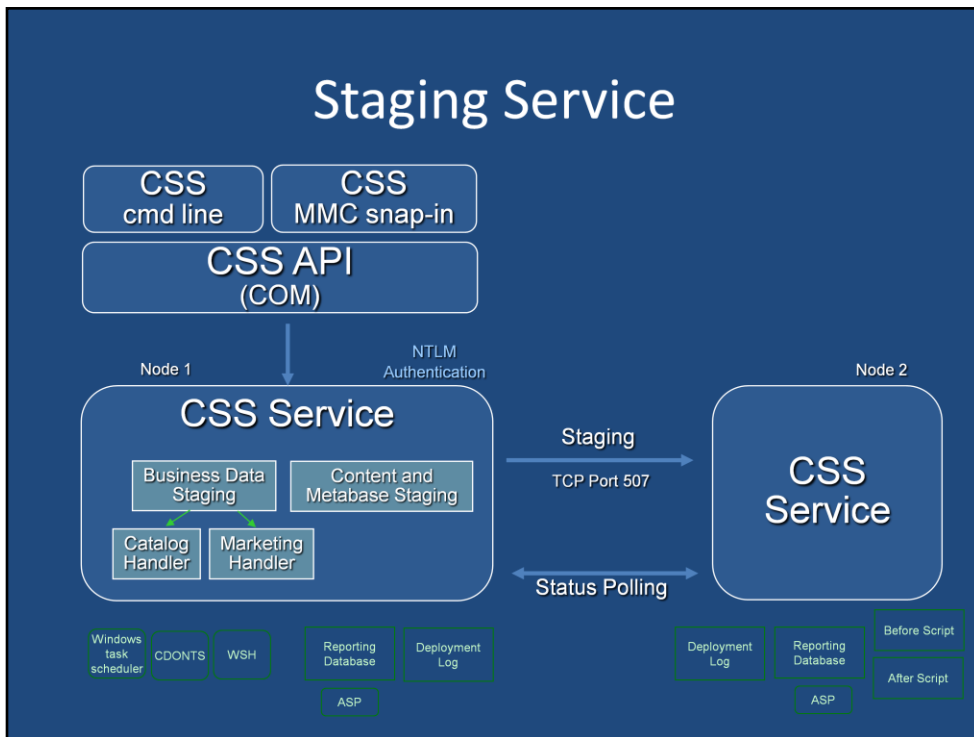


The Commerce Server staging Service, or CSS, provides content replication services for files and folders, IIS6 metabase settings, and Commerce Server business data including product catalogs, marketing campaigns, site terms, and orders configuration data such as payment methods and shipping rates. Each of these three project types requires a separate project to be configured, so for example if you're going to stage both site content and Commerce data, you need to create two separate staging projects for this.

The staging service uses the in-process mode of the Commerce Server API's to export commerce business data before replicating it and then again to import it on the destination. This is important to understand from a security perspective, since it is necessary to grant the CSS service account appropriate permissions to the Commerce Server SQL databases. The database roles required for this are documented comprehensively in the deployment section of the documentation.

Notice that a staging deployment may include multiple destinations, for example replicating site content out to each of the Web servers in a Web farm. A waypoint machine can be used to bridge domains where a direct connection isn't possible. Commerce Server business data is not imported on a waypoint machine but is simply passed through. Waypoint servers still require access to a SQL server since the install of CS requires a connection to an administration database and this also requires a full product license. A route in CSS is a reusable definition of a set of destinations which multiple projects can use and may also include the specification of credentials to use when authenticating against the destination.

Assuming that transactional staging of catalogs is not used and that virtual catalog capabilities are used judiciously, CSS can make updates to a production site without incurring downtime of the site. It's still wise to schedule staging pushes to occur during low traffic periods.



Staging includes both a command line interface and an MMC snap-in, either of which can be used to create and manage staging projects and to initiate replications. These in turn use the CSS COM API to communicate with the CSS service. Primary Interop Assemblies, or PIA's, are provided for .NET programmability against this object model.

The service itself uses TCP port 507 to service requests and to replicate content between nodes. NTLM authentication is used to authenticate client connections.

In order to report the success or failure of a staging job, the service uses polling to periodically check on the status of the active jobs at the project destinations until such times as the job completes or aborts. To support this, each node maintains a deployment log that contains the current status of jobs on that node. The deployment log is also used in conjunction with incremental business data staging. It maintains the time of the last successful export so that all the changes since that time can be exported incrementally and replicated to the destinations.

The system also maintains a reporting database at each node. The Reporting database is an Access database that stores a transaction history, sort of an audit log, for each job. The reports are exposed to users via a set of classic ASP Pages and are also surfaced through the MMC snap-in.

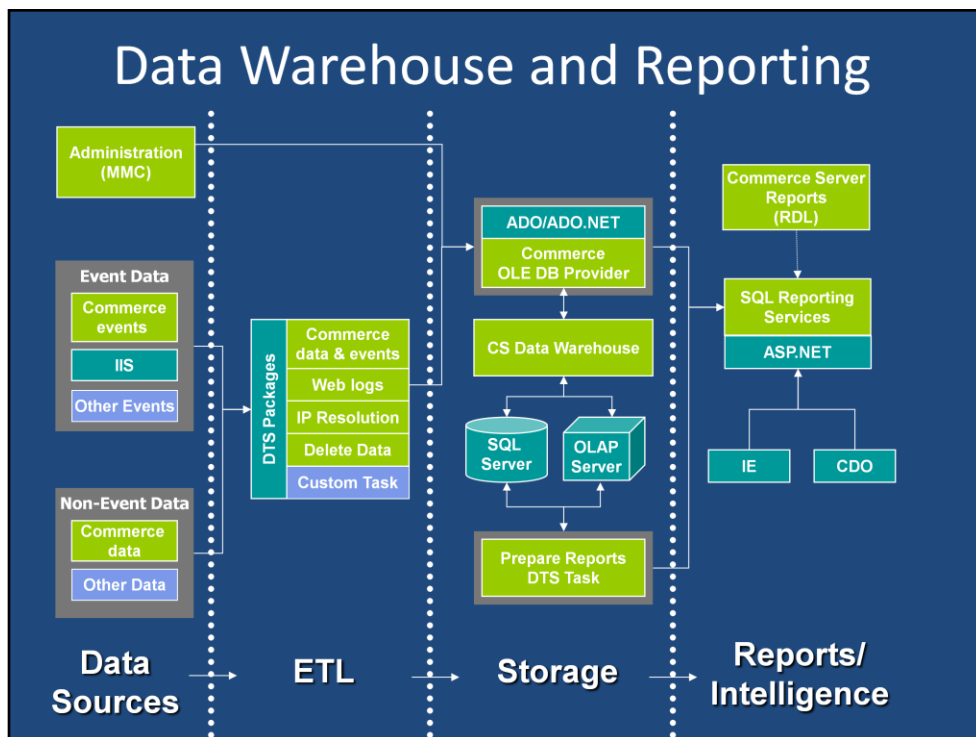
As part of the project creation process, scripts can be specified to run before and/or after a replication at both the source and destination endpoints.

The staging service supports scheduling of staging jobs through the Windows task scheduler. It also supports email notifications, which can be configured to be sent when a job completes, or when a job fails.

Project configuration for content projects is stored in the registry under the HKLM\Software\Microsoft\CSS key. For Business Data Projects, there is an additional XML configuration file created under the staging folder that is found under the Commerce Server installation directory. Windows ACLs are used to secure each project's registry key and XML file so that only authorized users, typically members of the CSS Operators or CSS administrators group can modify the settings.

Note that only CSS Administrators have permission to create projects. This is significant since if you use the Wizard to create a project, CSS will automatically attempt to create the project on all destinations specified or on all nodes in the route that's selected when you create the project. The credentials that are used to authenticate against the remote endpoint may be the credentials of the user, the credentials specified for the project, or the credentials specified for the route. Whichever credentials are in effect, the user that these authenticate as must be a member of the CSS administrators group on the remote machine in order for the creation to succeed. You can also manually create the identical project on the endpoint servers.

The final point here is on the architecture of the Business Data Staging driver. A business data handler exists for each of the four types of business data supported. If you open up the project XML file for a business data project, you will see the section that each of these handlers writes into the file complete with all of its configuration settings. This architecture will permit us to easily extend the system with support for new business data types in the future.



The final area we'll look at is the Data Warehousing and Reporting system. The Data Warehouse schema is largely unchanged from CS2002 but the reports are all new and written to use SQL Reporting Services.

Rather than report on data from the runtime databases directly, which can have severe consequences on runtime performance and the load on the runtime SQL servers, Commerce Server uses DTS tasks to Extract, Transform, and Load the data into a more denormalized schema to support efficient querying. This includes a highly tuned DTS task for parsing and importing the IIS log files from across a server farm. This task stitches together and merges the log files and provides services such as infencing of user visits. The logs may also contain commerce events such as marketing campaign events and basket events, which together with user identification cookies get imported from the logs into the Data Warehouse.

Like the rest of Commerce Server, the Data Warehouse supports both SQL Server 2000 and SQL Server 2005. If you are using SQL 2005, you need to install the backward compatibility module for DTS. This is a separately downloadable component and is not included with the SQL Server 2005 distribution.

The bulk loading of Commerce Server data is supported through the fastload path of the Commerce Server OLEDB provider. This again provides schema mapping services but this time using the Data Warehouse catalog.

Once the event data and Commerce Business data has been bulk loaded into the data warehouse tables, OLAP cubes can be generated. This task is carried out by the Prepare Reports DTS task. Since DTS tasks are nothing more than COM objects implementing a specific interface, these tasks can be scripted or scheduled to be run at regular intervals using the SQL server agent service.

The final step is to generate the reports and expose them to your business users. Commerce Server includes a report deployment tool that you use to install the canned RDL's to your reporting server system. This tool is called ReportInstaller.exe and is found in the tools folder off of the Commerce Server root directory. The reports can of course be customized to your sites own unique business requirements and can also be scheduled to run at fixed times and then exposed to your users. Some of the out-of-box reports run against the OLAP cubes while others access the relational data tables in the warehouse directly.

Resources

- Commerce Team blog
<http://blogs.msdn.com/Commerce>
- Commerce Server Forums on MSDN
<http://forums.microsoft.com/MSDN/ShowForum.aspx?ForumID=1059&SiteID=1>
- Commerce Server Home Page
<http://www.microsoft.com/CommerceServer>
- Developer Screencasts
<http://www.microsoft.com/technet/prodtechnol/comm/2007/webcasts.msp>