At a glance:

- Content lifecycle management with SharePoint
- Creating document information panels
- Editing Office and SharePoint documents via InfoPath

Standardise data management with custom content types

Pav Cherny

The large number of documents and other content items typically found in an enterprise environment pose business and technical challenges for managing the documents, their metadata, and their behaviours in a centralised and reusable way. Microsoft Office SharePoint Server 2007 promotes company-wide collaboration by allowing diverse teams within an organisation to share workspaces on web sites, document libraries and lists.

SharePoint 2007 makes it possible to standardise many aspects of content and lifecycle characteristics through content types. Site content types are metadata definitions that can be established independently of any specific site collection, site, list, or document library. This enables you to establish company-wide properties, workflows, information management policies and other elements consistently, while also allowing individual departments or site owners to customise content types for specific purposes.

In this article, I will show you how to use the new SharePoint content model introduced with Windows SharePoint Services (WSS) 3.0 and Microsoft Office SharePoint Server (MOSS) 2007 to build hierarchies for enterprise content based on general characteristics. These content hierarchies enable uniform application of metadata, workflows

and information management policies at a global level, while also providing the flexibility to accommodate unique content management needs at the level of site collections, sites, document libraries and lists.

To illustrate some of the low-level aspects of SharePoint content types, I have included a number of custom tools in the companion material, and I have also included the source code in case you want to extend these tools according to your specific needs. Just keep in mind that these tools have not been tested thoroughly and should not be used on a production system. You can download the code from the *TechNet Magazine* web site at www.technetmagazine.com/code08.aspx.

Content lifecycle and content type definitions

There are many details to consider when managing documents and other content in an organisation. Among other things, it is necessary to define which person or process needs to perform what action as content is created, published, archived and disposed. It is often also necessary for an organisation to develop specific templates for content creation, define roles to assign responsibilities and access privileges to users, provide version control, monitor compliance, store and archive, define metadata and so forth.

However complex a particular content lifecycle may be, the SharePoint content model recognises that there are some general characteristics and typical phases that determine how the individual content items should be handled. For example, you can structure content creation through templates and input forms, and content display and searches through metadata. You can also use editing, approval, or other workflow requirements, archiving requirements, expiration timeframes, and applicable information management policies to distinguish individual content pieces. Some content might not require any special templates or might never be archived, but even these are lifecycle characteristics that proceed to distinguish these items from other items.

The SharePoint content model allows you to define individual content types and establish hierarchical relationships. In a hierarchical relationship, children inherit general characteristics from parent content types, adding specific characteristics as needed.

The best way to illustrate this is to examine the built-in content types of SharePoint. WSS 3.0 and MOSS 2007 include a number of predefined content types for typical items that can be stored in a document library or list, including documents and tasks. You can find the definitions of these standard content types on a SharePoint server in the %CommonProgramFiles%\Microsoft Shared\Web Server Extensions\12\Template\Features\Ctypes folder. There you'll find a manifest file called feature.xml. Looking at this file, you can see that SharePoint implements the standard content type definitions as a hidden feature (Hidden="TRUE") with a site collection scope (Scope="Site") and that the Element manifest file, which contains the actual content type elements, is ctypeswss.xml (<ElementManifest Location="ctypeswss.xml" />).

If you are interested in learning more about SharePoint features, I recommend you read Ted Pattison's Office Space column in *MSDN® Magazine* titled "Features for SharePoint", which is available at: www.msdnmagazine.com/issues/07/05/OfficeSpace.

Now, let's open the ctypeswss.xml file in Notepad to examine the standard content types regardless of their visibility in the SharePoint user interface. You should not modify ctypeswss.xml. If you're thinking of editing ctypeswss.xml to add new fields to standard content types or to make hidden content types visible so that SharePoint users can use them to derive new content types, note that this is usually not necessary. Plus, it leads to unsupported configurations, and later installation of service packs may overwrite your customisations and break your content management solutions.

A much better approach is to copy what you need into a new element manifest file, add your customisations as necessary, and then deploy your custom content types as a new feature with a site collection scope so that they are available to all sites within the site collection.

The Collaborative Application Markup Language (CAML)-based definition of the System content type as specified in ctypeswss.xml is revealed here:

The SharePoint content model allows you to define individual content types and establish hierarchical relationships

```
<ContentType ID="0x"
    Name=$Resources:System
    Group="Hidden"
    Sealed="True"
    Version="0">
   <FieldRefs>
      <FieldRef ID="{c042a256-787d-4a6f-8a8a-
cf6ab767f12d}" Name="ContentType"/>
   </FieldRefs>
</ContentType>
```

The Group and Sealed attributes show that the System content type is hidden and sealed so that it cannot be changed in the SharePoint user interface. The System content type has only one FieldRef element that references a built-in site column called ContentType. This is the highest level of abstraction. All other SharePoint content types inherit this property from System content type. In this way, SharePoint ensures that all content items stored in any document libraries or lists have this property in common.

Figure 1 shows you a ContentTypeHierarchy Web Part, included in the companion material for this article, which illustrates the hierarchies more intuitively. System is at the root of all other content types, followed by Item, and so forth. The Item content type, for example, establishes the next level of detail. If you check ctypeswss.xml, you can see that Item defines a single metadata field that references a site column called Title. In that way, all built-in content types at lower levels have a Title property.

It is also possible to remove an inherited field, as the Document content type definition in ctypeswss.xml demonstrates. The Document content type includes several corresponding RemoveFieldRef elements, yet you might want to keep the Title field in place in your custom content types because the Title column provides access to the Edit Control Box (ECB) dropdown menu in SharePoint document library and list views.

A good example that illustrates how to rename inherited fields is the Far East Contact content type in ctypeswss.xml. Search for 0x0116, which is the corresponding content type ID, and then check the FieldRef element with the attribute Name="Title" to see how you can use the DisplayName attribute to rename a field in the user interface. In this particular example, the DisplayName attribute changes the name of the Title field in the user interface to a localised data value referenced by "$Resources:core,Last_Name;".

If you take a closer look at **Figure 1,** you can see the ID attribute of the ContentType element uniquely identifies the content type and establishes the hierarchical relationship. All IDs begin with the ID of the parent content type appended with additional hexadecimal values. Standard content types typically have two additional hexadecimal values appended to create a new unique ID for the child content type. Another technique is to append a "00" and a hexadecimal GUID. That's how the Office Data Connection File and Universal Data Connection File content types derive from the Document content type, for example.

An important point for users to remember is that the ID attribute of the ContentType element cannot be longer than 1,024 characters. This can turn out to be a problem in a large hierarchical relationship if all content types use the hexadecimal GUID addressing technique. However, it is not a good idea to start with the shorter technique because these IDs may not be unique.

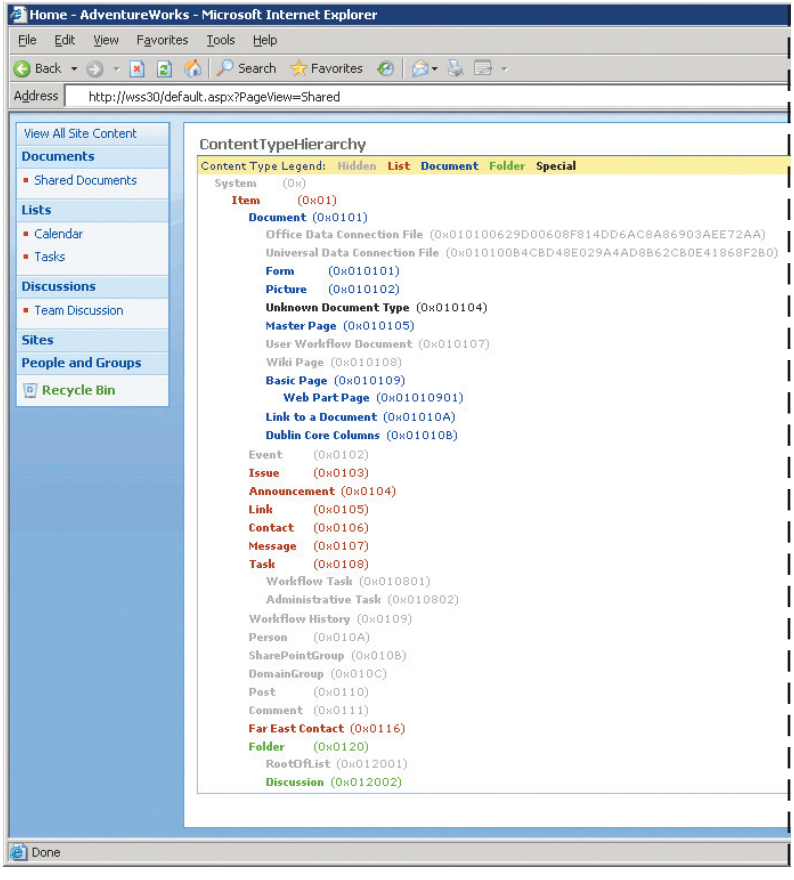

Figure 1 **WSS 3.0 built-in content type hierarchy**

To ensure uniqueness, use the GUID technique to establish a global namespace for your enterprise content types and switch to the shorter technique within that namespace. For detailed information about the ID attribute and other elements of content type definitions, see the topic "Content Type Definition Schema" in the WSS 3.0 SDK.

Content type dependencies

Now let's turn to the question of how to use SharePoint content types to structure the management of content items. You need to consider several dependencies, such as the interdependencies among document libraries and lists, content types, site columns and field types. Libraries and lists reference content types, which reference site columns, which in turn reference field types (such as the standard WSS field types text, note, choice, number, currency, and so forth), which in turn reside in Microsoft .NET Framework assemblies, such as the WSS core assembly Microsoft.SharePoint.dll.

For an example, let's return to the System content type as illustrated earlier in the CAML definition of the System content type manifest file entry. As you saw, this content type contains a FieldRef element that refers to a site column called ContentType. Note that the content type definition does not define the actual ContentType site column. ContentType is a Text field, defined in the Elements manifest fieldswss.xml, located in the %CommonProgramFiles%\Microsoft Shared\Web Server Extensions\12\Template\Features\Fields folder. The Text field type, again, is defined in fldtypes.xml, which you can find under %CommonProgramFiles%\Microsoft Shared\Web Server Extensions\12\Template\XML. An important point to take away from this dependency tree is that you must ensure that all of the resources are available on your SharePoint servers.

The content types you want to use must be created at the site level of your document libraries and lists or higher up in the site hierarchy. Similarly, the metadata fields of your content types must refer to existing site columns. You can add standard or custom site columns based on standard or custom field types to your content types, but you must ensure that the site columns exist on the local site. Furthermore, if you want to use custom field types, you also need to ensure that the underlying .NET assemblies are deployed on your SharePoint servers.

Similar dependencies exist for content types that reference document templates, custom event receivers, workflows, and other components. For example, the content type definition can contain a DocumentTemplate element that points to the document template associated with the content type. The content type definition can also contain an XmlDocuments element with one or multiple XmlDocument child elements that define additional characteristics of the content type, such as namespace definitions, document information panel definitions, or any custom information.

The content types must be created at the site level of your document libraries and lists

Establishing global content types

Users with authoring permissions can create new content types and site columns directly in the SharePoint user interface, but the content types are then only available in the local site and below in the site hierarchy. Custom

SharePoint resources

- **InfoPath Team Blog**
  http://blogs.msdn.com/infopath
- **Workflows in Office SharePoint Server 2007**
  http://msdn2.microsoft.com/ms549489
- **TechNet Webcast: Installing and Configuring Search in SharePoint Server 2007**
  www.microsoft.com/uk/configuringsearchwebcast
- **InfoPath Forms Services**
  http://msdn2.microsoft.com/ms540731

site columns are only available in the local site. This is not sufficient if you want to establish global content types. You need to ensure that global content types are available across all site collections in your environment. This is where SharePoint features come in handy. It is straightforward to install and activate SharePoint features across multiple site collections to apply the corresponding site column and content type definitions consistently across all locations.

The companion material for this article includes a sample feature called AdventureWorks that demonstrates how to establish a global content type. The feature defines a general content type called Customer Documentation that can be used to create any type of customer materials such as proposals and sales letters. It is reasonable to assume that all these types of documents should be associated with customer information such as company name, contact details, and address. I specifically created a custom field called Customer Name for your content type and added some built-in fields such as Department and Primary Phone. You can change the content type and fields by editing the ContentTypes.xml file and the field references by editing the SiteColumns.xml file in the companion sample.

If you deploy the feature as described in the readme.htm file, you can activate it consistently across multiple site collections. The Customer Documentation content type is then available globally. With that, the individual departments can create specific customer documentation through derived content types associated with targeted document templates. The companion material includes sample document templates that could be used in sales and consulting. **Figure 2** shows the resulting content types.

You have several options to choose from to create features for custom site columns and content types in WSS 3.0 and MOSS 2007. You can study the WSS 3.0 SDK and write XML files from scratch. You can also choose to use SharePoint Designer to export a list into a personal Web package, rename the resulting .fwp file into a .cab file, extract the manifest.xml file from the .cab file, and then search for ContentType definitions in the manifest.xml file. Unfortunately, both approaches are inconvenient and error prone.

In contrast, it is remarkably easy to create custom site columns and content types in the SharePoint user interface. Yet, the interface does not provide an option to edit the XML files of a feature. Features are an efficient way to provision SharePoint resources, but once provisioned the resources only exist in the content database. Perhaps a future version of WSS will include the ability to export site columns and content types into XML files through the SharePoint user interface, similar to exporting Web Parts. For now, you need to work with what's here.

The companion material includes a Web Part called ContentTypeSchema that can serve as a starting point. It uses the SharePoint object model to extract the SchemaXML property from the selected content type. Through eXtensible Stylesheet Language Transformation (XSLT)-based transformations, the Web Part derives Field definitions or ContentType definitions depending on the option you select in the user interface before you click on the Display button.

Figure 3 shows this Web Part in action. It displays the Active Directory Documentation

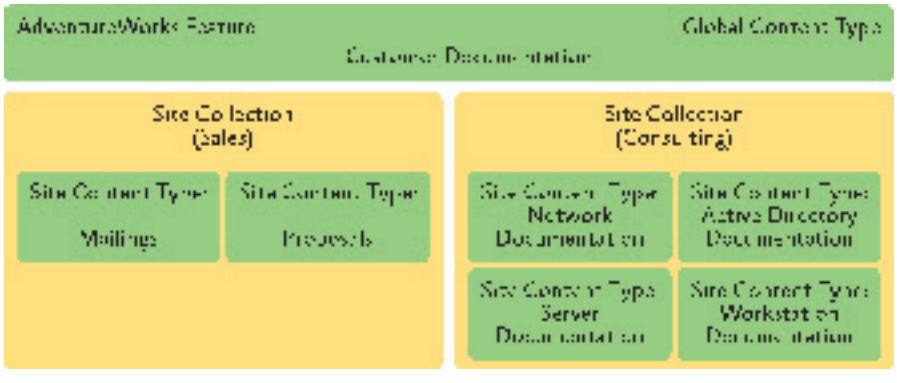

Figure 2 **Parent and child content types for customer documentation**

content type, which I based on the Customer Documentation content type. The Active Directory Documentation content type is associated with a custom Microsoft Word template (Active Directory Template.dot). Note that this content type has no additional fields. All fields are inherited from the parent content type.

If you are planning to use the ContentTypeSchema Web Part in your own environment, keep in mind that this Web Part has not been tested thoroughly. My Web Part uses relatively complex XSL transformations to build a delta between the SchemaXML property of the currently selected content type and its parent content type. Yet, XSLT 1.0 is not really designed for advanced XML document conversions. There is no built-in feature to compare XML nodes. Also, XML namespaces, as well as CDATA sections, represent difficulties that XSLT 1.0 cannot handle efficiently.

SharePoint stores the definition of site columns and site content types created by using the SharePoint user interface or SharePoint object model in the content database in a table called ContentTypes. In **Figure 3**, take a look at the ID of my custom content type. Accordingly, the following T-SQL statement would deliver exact results: SELECT Definition FROM ContentTypes WHERE ContentTypeId = <content type ID>.

Whichever approach you decide to use, creating features for enterprise-wide content types is very straightforward once you've obtained the site column and content type definitions. I recommend that you use a single feature for all global site columns and content types of your department or company. That way, it is clear where to add new site column and content type definitions.

Enterprise-wide searches on custom metadata

An immediate advantage of content type hierarchies is that all child content types inherit the parent content type's metadata fields. Because all content types have metadata fields in common, it is quite easy for users to create custom views and filters in the document libraries.

The AdventureWorks sample feature demonstrates this quite intuitively. Regardless of the content that a consultant or salesperson creates, saving the resulting Word 2007 document in the document library requires specifying a Customer Name because the Customer Documentation parent content type defines this metadata field as required. By grouping or filtering the items in a document library view according to the Customer Name, consultants and sales team members can locate all of the existing documentation for a customer quickly and conveniently, as shown in **Figure 4**.

Common metadata also makes it easy to locate content across multiple document libraries and sites by using the search capabilities

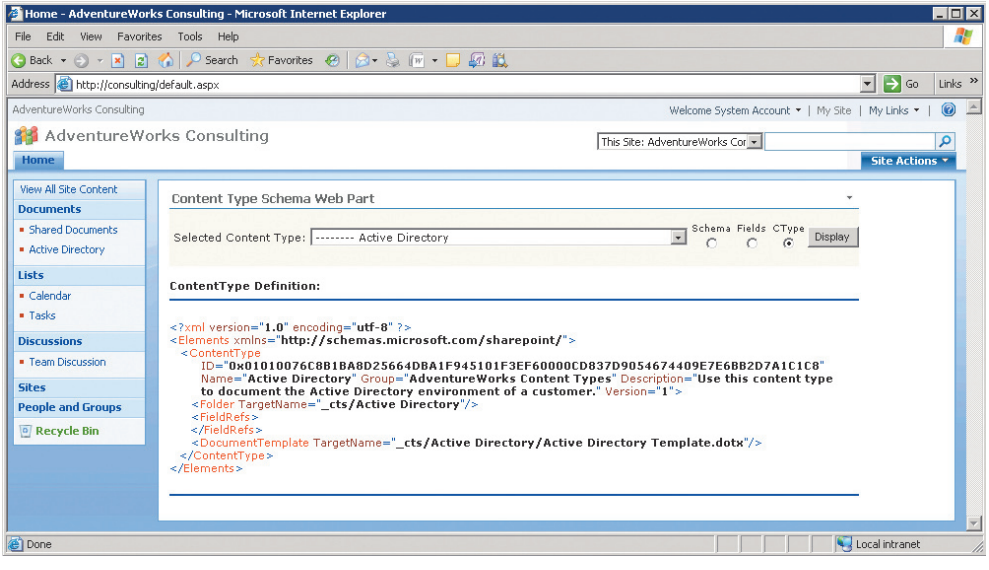

Figure 3 **Custom content type definition in the ContentTypeSchema Web Part**

of WSS 3.0 and MOSS 2007. WSS supports searches across libraries, lists, and sites within a single site collection. MOSS 2007 goes beyond these basic capabilities by allowing you to implement an enterprise-wide Search Centre and to manage custom metadata in the SharePoint 3.0 Central Administration user interface. For this reason, the following explanations focus on MOSS 2007.

When you configure a Shared Service Provider (SSP) in MOSS 2007 and crawl your SharePoint sites, MOSS 2007 can automatically discover the custom metadata fields used in the content sources. Accordingly, you can find the custom metadata fields in the list of crawled properties.

Metadata fields defined in SharePoint features usually fall under the SharePoint category. To find its location in SharePoint Central Administration, on the Quick Launch menu under Shared Services Administration, click on the link to your SSP, click on Search settings, click on Metadata property mappings, then click Crawled Properties on the Quick Launch menu, and then proceed to open the SharePoint category.

I'll give you an example of this in action. The metadata field named Customer Name ends up as a crawled property called ows_Customer_x0020_Name. SharePoint prefixes crawled properties with "ows_", and "_x0020_" is the encoded version of a single space. If you display the details of this crawled property, you can find that it is included in the search index by default so that users can employ the values of this crawled property to perform content searches. However, to increase the search precision, you might not want to map the crawled property to a managed property so that users can explicitly search for content by customer name.

You have two options when mapping crawled properties to managed properties. You can automatically generate a new managed property for each crawled property or you can map crawled properties manually to managed properties. The first option is available when you display the settings of the desired crawled property category (when displaying the crawled properties in a category, such as the SharePoint category, click on the Edit Category option on the Quick Launch link). Under Bulk Crawled Property Settings, you would have to select the checkbox "Automatically generate a new managed property for each crawled property discovered in this category." However, SharePoint prefixes automatically created managed properties with "ows" and escapes spaces with "x0020". The managed property for the crawled property ows_Customer_x0020_Name would be owsCustomerx0020Name.

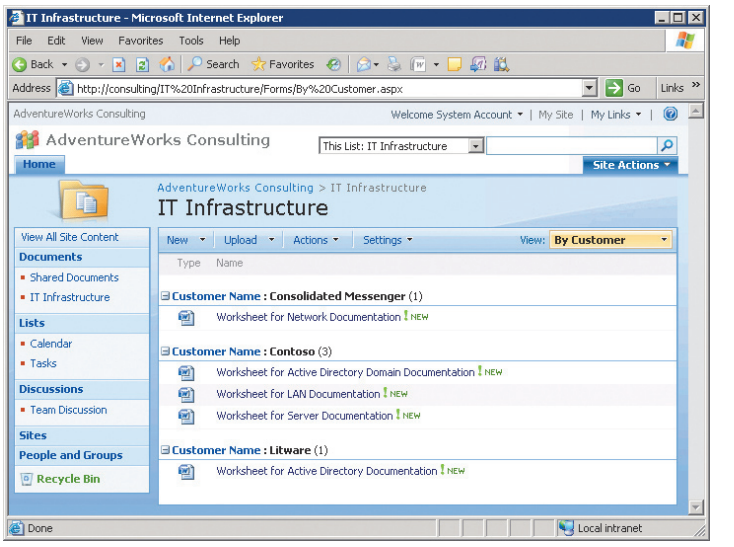

Figure 4 **Grouping various documents in a library based on common metadata**

An InfoPath form provides a convenient way to edit the properties of a SharePoint content type

This, however, is not a very user-friendly property name.

Perhaps a better strategy is to map crawled properties to managed properties manually after you deploy your enterprise-wide content types. You can map a crawled property to one or multiple managed properties. To create new managed properties, in SharePoint Central Administration, on the Quick Launch menu under Shared Services Administration, click on the link to your SSP, click on Search settings, then Metadata property mappings, and then click on New Managed

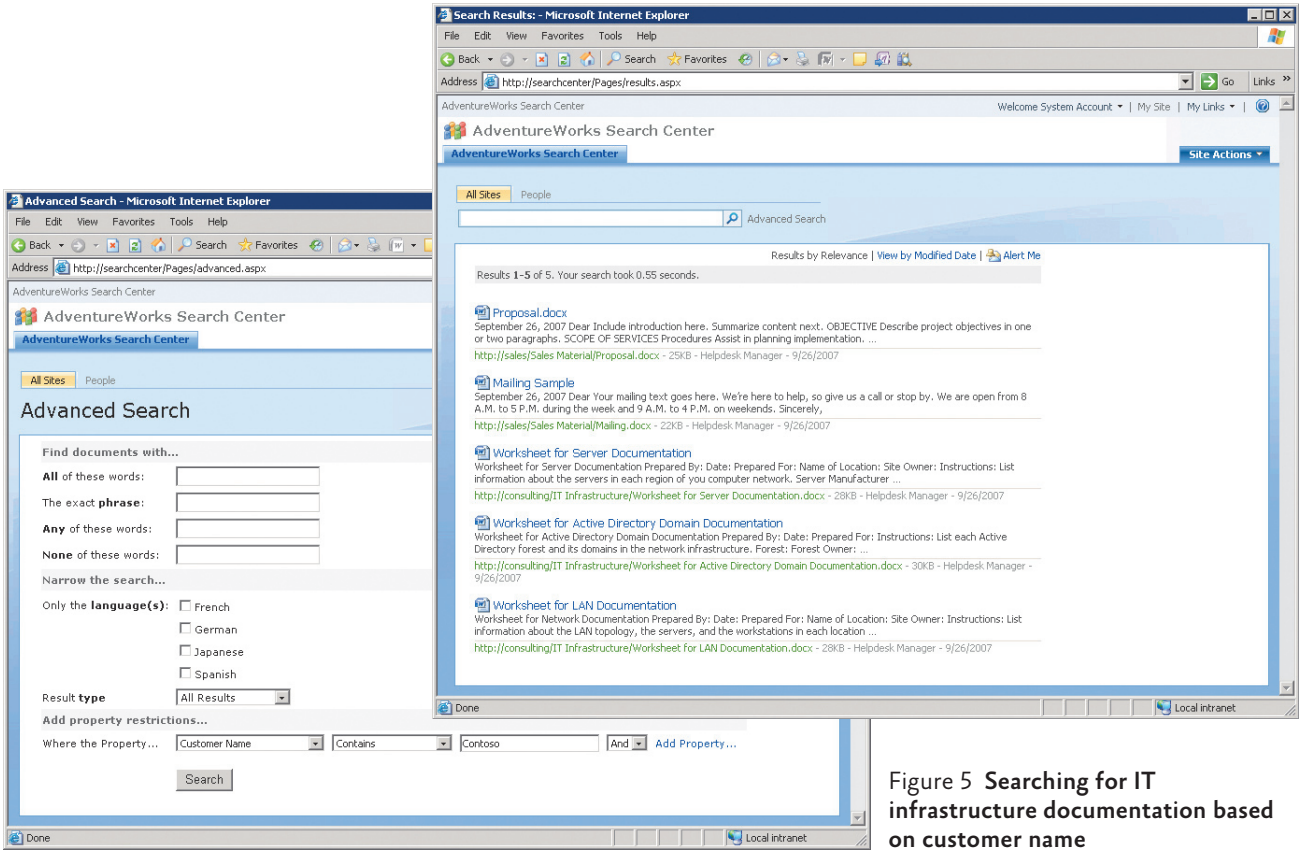

Figure 5 **Searching for IT infrastructure documentation based on customer name**

Property to specify the required information and associate the new managed property with the desired crawled property.

Users can then locate relevant content items by using the managed properties either in property: syntax or by using advanced search. For example, if you map the crawled property ows_Customer_x0020_Name to a managed property named Customer, users can then search for all documents of a customer simply by specifying Customer: <Customer Name> in the standard search box, such as Customer: Contoso.

It is also a good idea to include managed properties for the most important metadata fields of your content types in the properties list on the Advanced Search page. To accomplish this, display the Advanced Search page, then click on-site Actions, and select the Edit Page command. Now you can click Edit on the Advanced Search Box to select the Modify Shared Web Part option. When you expand the Properties category and place the cursor into the corresponding text field, you can click on the button to edit the Properties XML document. You need to add a property definition to the <PropertyDefs> element, such as <PropertyDef Name="Customer" DataType="text" DisplayName="Customer Name"/>, and you also need to add a reference to this property definition under a ResultType element (for example, the element <ResultType DisplayName="All Results" Name="default">), such as <PropertyRef Name="Customer" />. **Figure 5** displays the resulting Advanced Search user interface.

Ensuring information consistency

At this point, I have successfully standardised the most important metadata fields and content types and extended search capabilities across site collections in my enterprise based on MOSS 2007. Now it's important to ensure that users enter accurate information into the metadata fields.

There are two ways to do this. First, you can replace the standard document information panel in your document templates with a custom InfoPath® form that provides your users with a predefined selection of input options, such as you have in a listbox. Second, you can bind an event receiver to the con-

tent type and then check the accuracy of metadata and other information whenever users create, modify, or delete corresponding content items.

Both options complement each other. An InfoPath form primarily provides a convenient way to edit the properties of a SharePoint content type, whereas an event receiver can ensure valid metadata even if users edit the content type properties outside the InfoPath form. You can bind event handlers to a specific content type, which enables you to specify events and their responses for all documents associated with an individual content type in all site collections, regardless of the document library. You can find more information on how to develop and deploy event handlers at http://msdn2.microsoft.com/ms453149.

Perhaps the easiest method for associating a content type with a custom document information panel is to display the Document Information Panel settings of the content type in the SharePoint user interface on a computer with InfoPath 2007 installed. You can then click on the Create a New Custom Template link under Document Information Panel Template to start InfoPath 2007 in design mode. However, if your site content type includes site columns without an explicit SourceID attribute, you might encounter a situation in which InfoPath cannot create a valid XSD schema for the Document Information Panel form. For example, the Customer Documentation content type introduced earlier includes several contact-specific columns such as Department, Office and email that can cause this problem, as illustrated in **Figure 6**.

At this point, you have two options should you encounter this problem. You can either remove references to site columns without an explicit SourceID attribute from the content type definition, or you can replace the built-in site columns that cause the problem with custom site columns that are compatible with InfoPath 2007. Keep in mind that you cannot change the field references of a CAML-based content type after it was provisioned into the content database, especially if you already created child content types. You cannot simply update the CAML-based content type definition file anymore, nor can you push down changes to child content types because Windows SharePoint Services does not track changes made to the parent content type definition.

To push down changes, you must change the parent content type through the SharePoint user interface or object model, or you

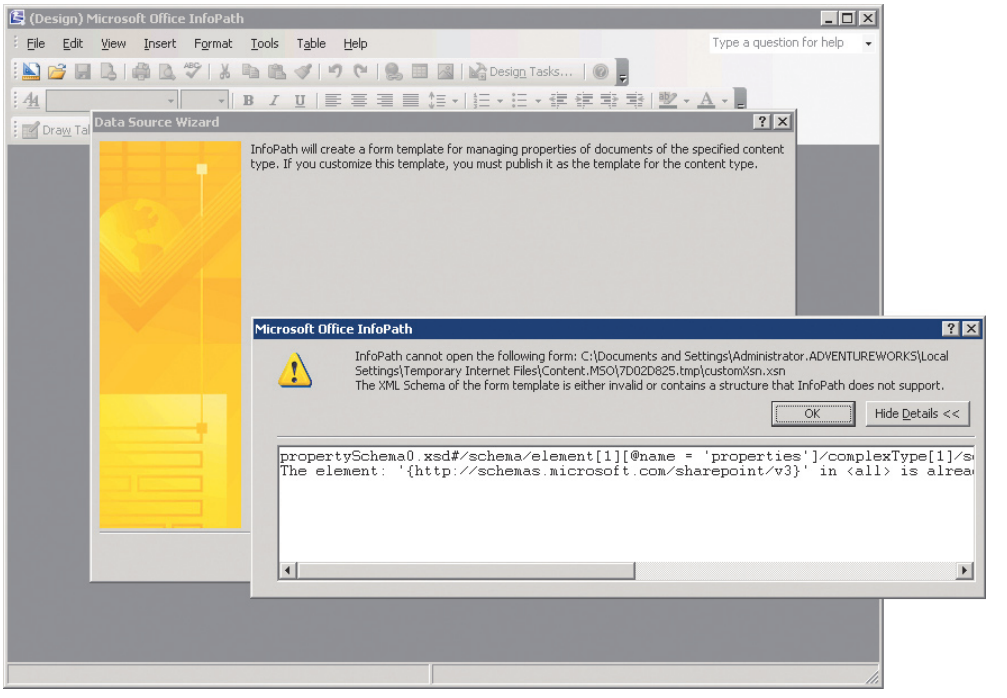

Figure 6 **XSD schema incompatibilities in InfoPath 2007**

must define a new content type that is derived from the existing one. The latter technique enables you to remove the critical fields and add new columns. Your users can then derive future content types from the new content type. To prevent users from choosing the wrong content type, add the previous content type to the _Hidden content type group.

As I said before, you cannot update CAML-based content types after deploying and activating them. Thus, it is very important to test global content types before deployment. For more information, see the MSDN article "Updating Content Types" at http://msdn2.microsoft.com/aa543504.

Once you have created a content type with proper field references, you can create a custom document information panel in InfoPath 2007. The best strategy is to let site owners create custom document information panels for their child content types. InfoPath 2007 provides the option to publish the custom document information panel directly to the selected content type, which facilitates the deployment scenario. Yet, it is also possible to publish the InfoPath form at a central location, such as a document library, and include a reference to the document information panel in the content type. This is the best option if you plan to provide a custom document information panel along with your CAML content types. **Figure 7** illustrates the implementation architecture.

The companion download for this article includes a feature called AdventureWorks_Update that extends the previous AdventureWorks feature by adding additional site columns that work with InfoPath 2007. The AdventureWorks_Update feature marks the original Customer Documentation content type as hidden and derives a new content type called Customer Docs, which replaces the inherited built-in fields with the new site columns and associates the new content type with a custom document information panel.

The new Customer Docs content type definition includes an XmlDocument element that provides information about the document information panel. Specifically, the xsnLocation element points to the InfoPath form http://companyresources/DIPs/customerDIP.xsn, which implements the document information panel. For detailed instructions on how to apply the AdventureWorks_Update feature, see the readme.htm file in the AdventureWorks_Update folder.

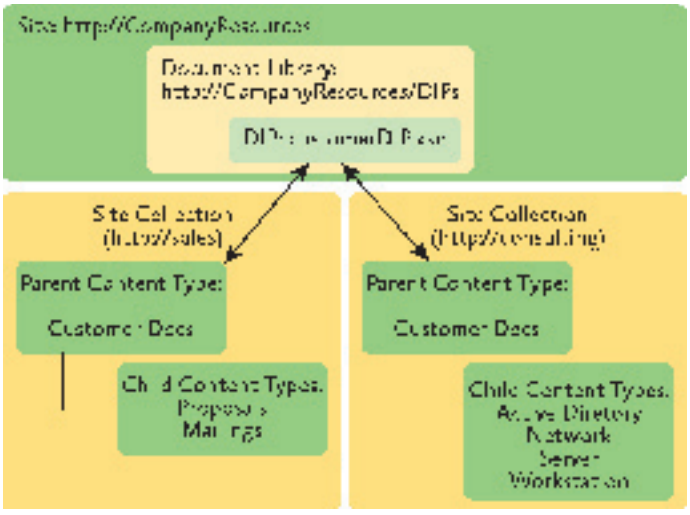

Figure 7 **XSN file implementation in a central location in MOSS 2007**

Wrap-up

You can use SharePoint content types to create metadata policies and use them consistently across the enterprise for content management. The hierarchy of content types enables you to standardise characteristics relevant to the entire enterprise environment and apply them uniformly to all sites through inheritance.

Among other things, it is possible to extend the built-in search capabilities of MOSS 2007 so that users can locate specific content faster and more conveniently. It is also possible to enforce information consistency related to metadata and establish centralised information management policies The best strategy is to standardise global content types on the most abstract metadata characteristics to minimise the need for later changes. Based on a carefully designed content model, SharePoint content types can provide new capabilities to standardise content lifecycle management across the enterprise. ■

For more information, visit the DPM TechCentre at: *http://technet.microsoft.com/en-gb/dpm/default.aspx*

Pav Cherny *is the president of Biblioso Corporation and an IT expert and author specialising in Microsoft technologies for collaboration and unified communication. His focus is on IT operations and system administration.*