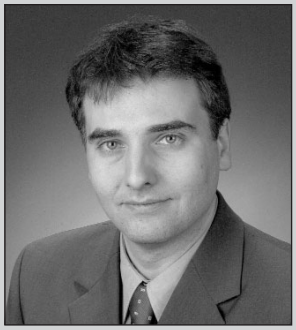
Consolidate and Integrate All of Your SAP and Non-SAP Documents, Transactions, Workflows, and Data with SAP Records Management

Joachim Becker



Joachim Becker is Product Manager of SAP Records Management. He joined SAP in 1995 as a developer, and also worked as a trainer and consultant. Since 1999, Joachim has assumed various product management responsibilities in SAP's Technology Development department. He is an expert in Business Process Technology, and has focused on the areas of Workflow, Document Management, and Communication.

(complete bio appears on page 60)

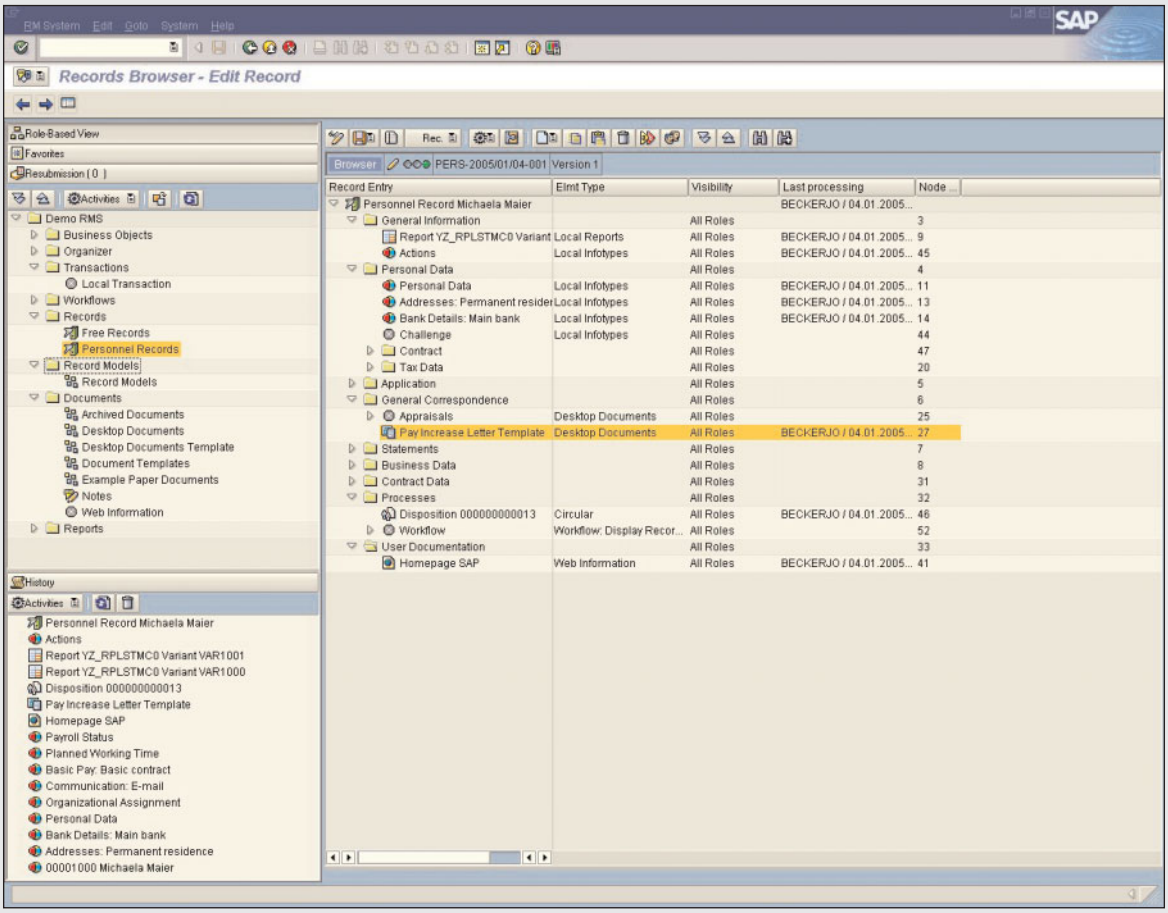
When it comes to managing and integrating documents and transactional data, most company landscapes resemble a bunch of loosely integrated "point" solutions with documents spread across myriad databases, file systems, and other data stores. For years the industry has been circling around a core requirement that no one could meet: a single, practical place where users can access all of their enterprise data, regardless of whether it's in an SAP R/3 transaction, a third-party system, a file system, a document management system, or an archiving system. The main roadblock was having a single system capable of integration with both SAP *and* external systems based on Java, Microsoft, and other third-party technologies — until now.

SAP Records Management, a standard component of SAP Web Application Server (SAP Web AS) 6.20 and higher (but licensed separately), offers the solution. It lets you integrate transactions, data, and documents from SAP *and* non-SAP systems, regardless of their module or vendor.

SAP Records Management is built on top of a new framework within SAP Web AS called the Service Provider (SP) Framework. The SP Framework leverages SAP Web AS's out-of-the-box ability to communicate with all types of external systems through protocols such as RFC, EDI, JDBC, and ODBC. Also, since it is hosted within an SAP system, SAP Records Management can accomplish what none of its predecessors could: bring SAP data, links to SAP transactions, and non-SAP content into a single view.[1] Plus, with a little custom development,

[1] Developed as a series of ABAP classes, the SP Framework provides a generic set of interfaces that applications such as SAP Records Management can use to connect to, manipulate, and integrate data and documents in SAP and non-SAP systems.

Figure 1 A Personnel Record Displayed in the Records Browser

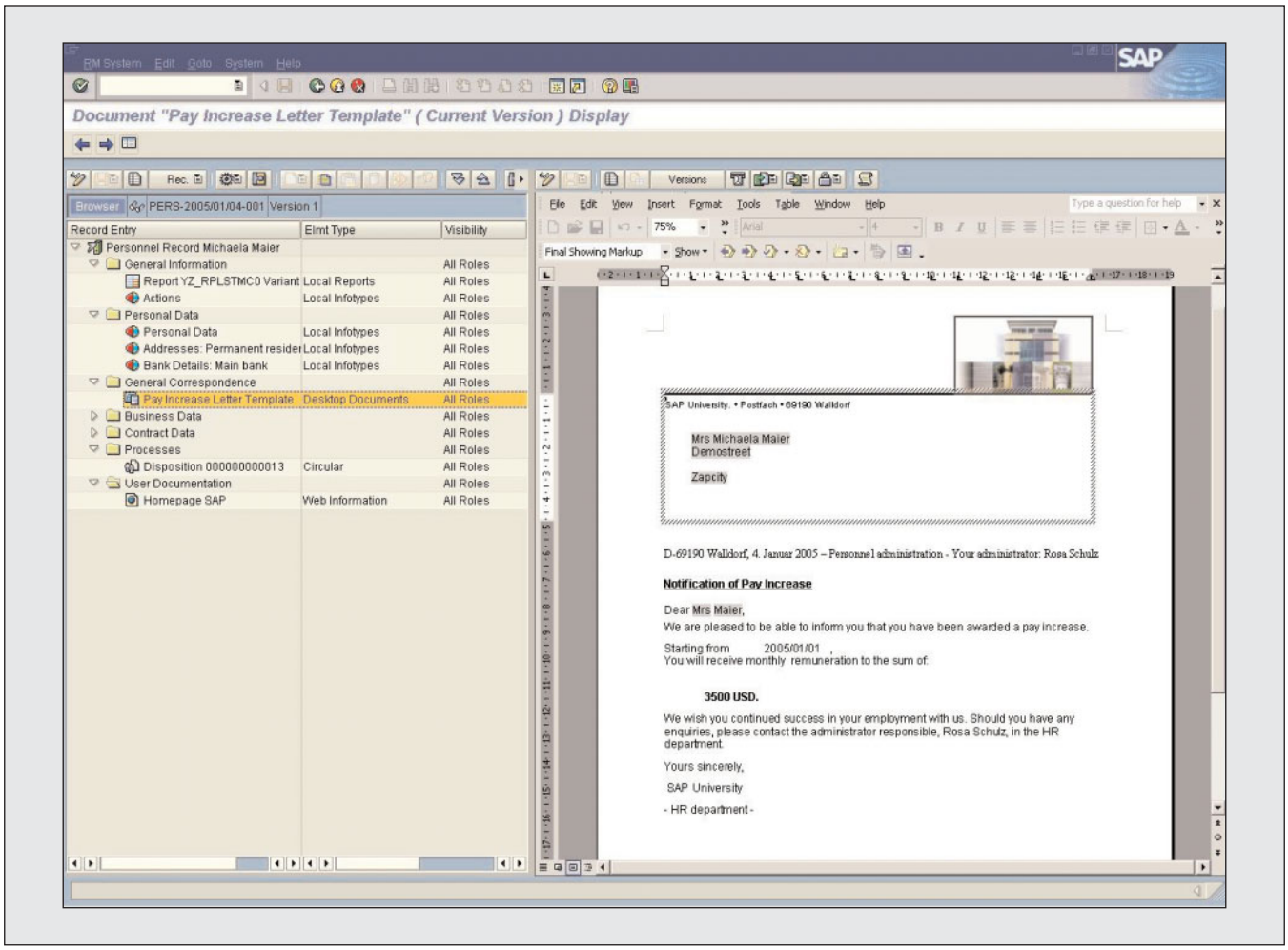


you can even add specific records from external databases to this consolidated view without having to bring the data into SAP — the data stays in the external database and is dynamically retrieved when requested by the user. Full integration with SAP ArchiveLink provides instant access to documents stored in file, archive, and content management systems.

Take a look at the HR example in **Figure 1**. It shows a personnel record for employee "Michaela Maier" displayed in the Records Browser, which together with the Records Modeler and Records Organizer constitute the key components of the SAP Records Management interface. The Records Organizer acts as a central desktop for maintaining records — it automatically launches the Records Browser (for viewing and editing records) and Records Modeler (for defining the structure of a record) as needed.

> ✓ ***Note!***
>
> *In the context of SAP Records Management, records can represent virtually anything, not just employee information. For example, records can contain links to documents and data relating to customers, vendors, or even "cases" representing customer complaints.*

Figure 2 Editing a Microsoft Word Document Within the Records Browser

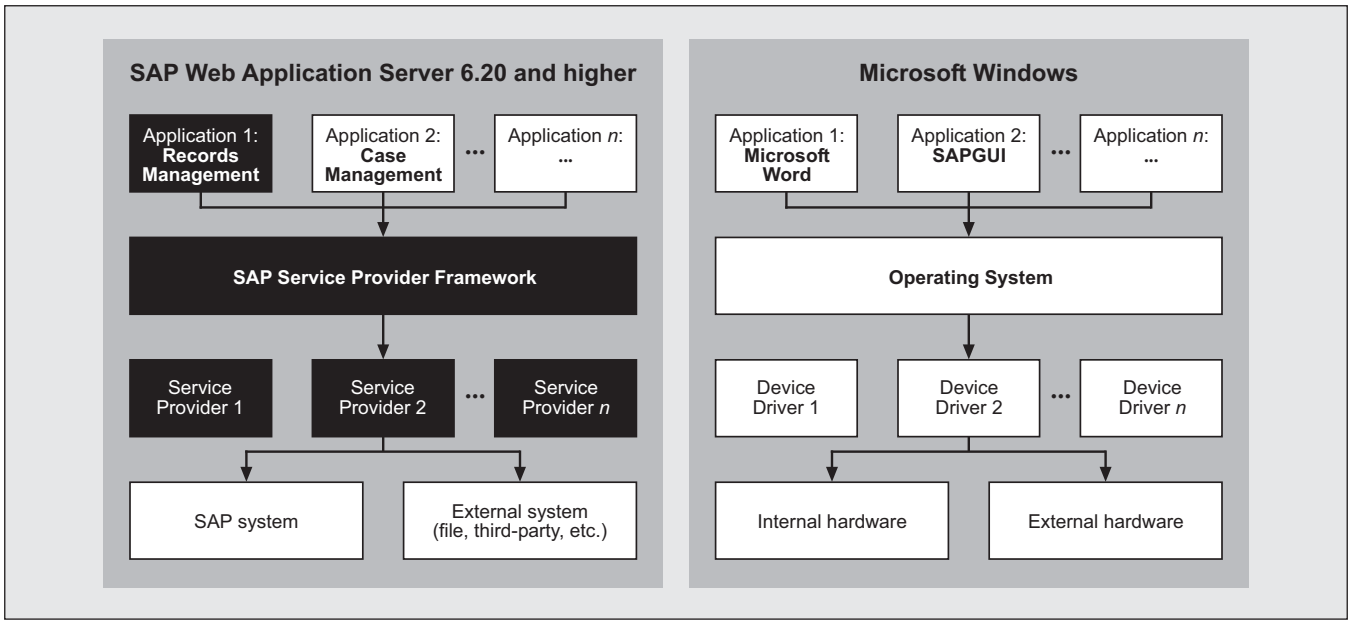


Notice in Figure 1 how the items on file for Michaela include infotypes from the SAP R/3 HR database (e.g., her address and bank details), correspondence documents, and links to relevant Web sites. All a user has to do is double-click on an item and the browser instantly brings up the transaction, document, or data in the right-hand pane of the SAPGUI. Clicking on the Pay Increase Letter Template item (under the General Correspondence folder) opens the document in Microsoft Word within the Records Browser display (see **Figure 2**). This truly makes the Records Browser an information hub for SAP users who work intensively with records, greatly improving their efficiency and reducing business errors resulting from "lost" records. Users no longer have to remember where information is stored, or spend time launching separate transactions or GUIs to access it.

But that's not all. You can also add workflows to Michaela's record and view the status of all workflows associated with her user ID — e.g., personal leave requests currently pending approval by her manager.[2] The Disposition item in Figure 2 (under the Processes folder) is an SAP workflow. In addition, you can add shortcuts to transactions in SAP and non-SAP systems, such as transaction PP03 to display all production orders Michaela has confirmed. Finally,

[2] Support for SAP workflows is built-in. You can integrate workflows from non-SAP systems by writing a custom service provider (for details, see the documentation section at http://service.sap.com/recordsmanagement).

Figure 3 SP Framework Components Are Analogous to PC Components



with a little bit of ABAP code,[3] you can manage, read, and modify records in legacy systems from within the Records Browser.

This article will give you the preliminary knowledge you need to evaluate and set up SAP Records Management. I will show you how to use the Records Modeler and Records Organizer to set up a simple example customer record using standard SAP-delivered service providers. I'll also show you how to access and manage records with the Records Browser. Finally, I'll discuss some common enhancements you'll probably want to make as a part of your implementation, such as Web access and workflow integration.

SAP Records Management Fundamentals

Before we can build our example customer record, there are a few terms and concepts you need to understand. I'll take you through them in the following sections.

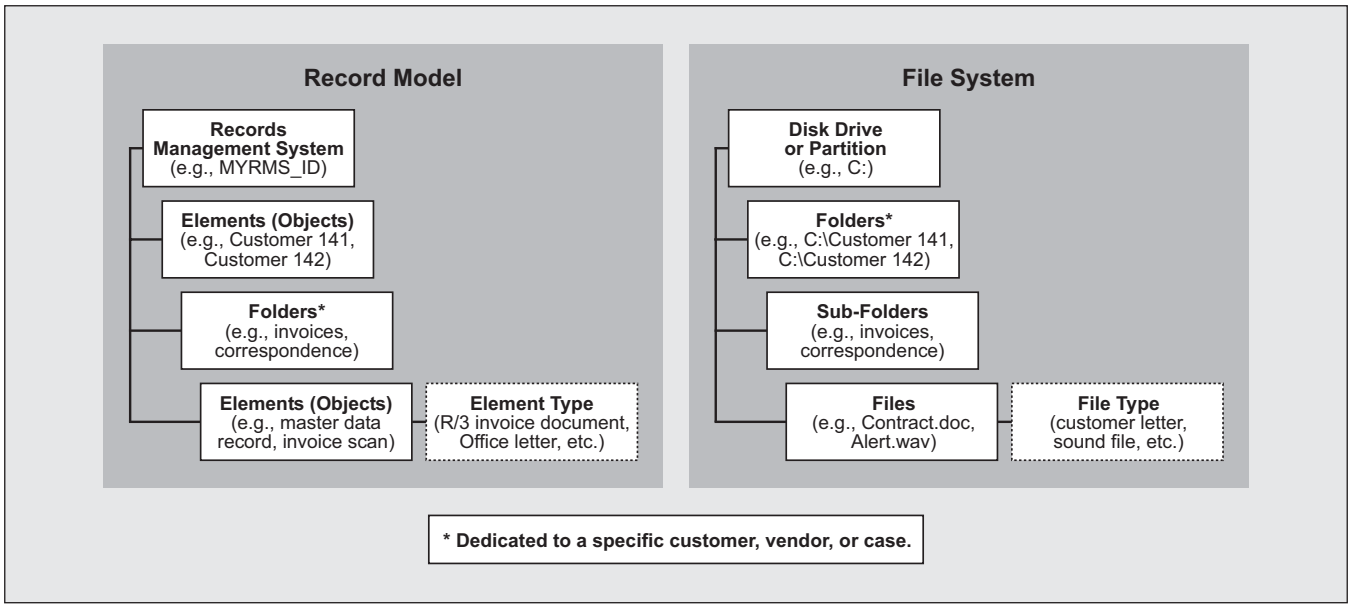
Service Provider (SP) Framework Components

As I mentioned at the beginning of the article, SAP Records Management utilizes a relatively new and powerful addition to SAP Web AS called the Service Provider (SP) Framework. The SP Framework enables SAP Records Management to efficiently manage vastly different objects — such as transactions, workflows, and documents — in a consistent way. As illustrated in **Figure 3**, the SP Framework is analogous to the operating system on your PC, which coordinates program access to hardware-based resources such as disk drives and printers. Similarly, the SP Framework coordinates access to backend systems, and provides applications such as SAP Records Management with a standard API to access these resources.[4]

[3] That is, by writing a custom service provider for the non-SAP system or database in ABAP.

[4] Note that the Case Management application shown in Figure 3 is used in solutions like mySAP Financials and mySAP Customer Relationship Management (CRM) to manage complex business processes such as disputes and service requests.

Figure 4 *SAP Records Management Objects*



To facilitate communication with backend systems, the SP Framework relies on *service providers*. Service providers are analogous to device drivers in Microsoft Windows (Figure 3). In Microsoft Windows, a driver provides a standardized API with which the operating system can connect to, read from, or write to a device. It handles the device-specific details internally. Similarly, the SP Framework defines a standardized API (technically, an ABAP Objects interface) with methods that service providers must implement for their respective systems. I will explain how to use standard SAP-delivered service providers later in this article.

Service providers have a specific user interface, and allow you to access the Business Content technical repository. The service provider for SAP ArchiveLink, for example, provides access to scanned originals on an external content server (an archive system or the standard SAP Web AS Content Server). For an overview of the standard service providers that come with SAP Web AS 6.20, see the sidebar on page 45.

SAP Records Management Objects

Figure 4 summarizes the core SAP Records Management objects, and extends the PC analogy used in Figure 3. A records management system segregates data, much like a disk drive or disk partition does on a PC. You create records management systems to group records that belong together, such as HR employee records with grievance case records.

Contained within a records management system are *records*. A record is an electronic "container" that groups links to *business objects*, such as customer master data, documents, transactions, and workflows. The record also contains placeholders for business objects that will populate the record in the future. Together, business objects and their placeholders are referred to as *elements*. When designing a record, each element must be described as a member of a specific *element type*, such as an SAP R/3 invoice object, a document (either plain text or a type of Microsoft Office document), a URL, etc. Element types map elements to the service providers that provide the access to the actual information objects in the backend system, and store important type-specific attributes (e.g., where logical system elements of that type are stored, into which folder the element should be placed in the Records Organizer tool, etc.).

> ✓ ***Note!***
>
> *A record itself can also be an element, so your records can include links to other records.*
>
> *English-speaking customers are sometimes confused by this use of the word "record"; the term is often used colloquially to refer to an actual document, not its container — e.g., "I asked for a copy of my medical records." This is easily resolved if you imagine that your doctor holds just one medical record for you that contains multiple objects, such as lab results, x-rays, etc.*[5]
>
> *Another potential point of confusion is that data in ERP and database systems is sometimes called records — e.g., "customer master record." Where the context is not clear, I use the terms "electronic record" and "business object" to distinguish between the two.*

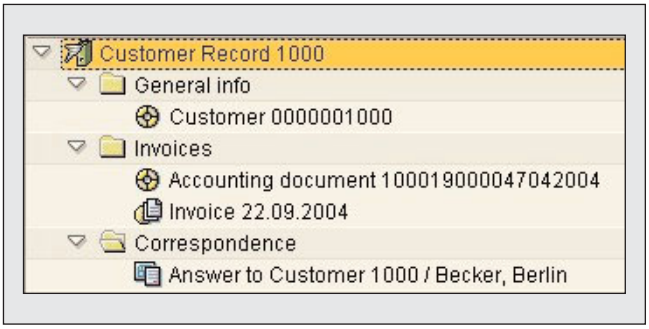
Records can also contain *folders* that organize and simplify the user interface. Creating a record is like creating a folder on your PC in which you can place shortcuts to data, documents, and functionality related to a particular customer or case, for example. For instance, our example customer record will contain three folders, as shown in **Figure 5**:

- A **General info** folder with a link to the customer's master data in SAP R/3
- An **Invoices** folder with a link to SAP R/3 invoices for the customer and associated scanned documents in an archiving system
- A **Correspondence** folder with correspondence to the customer (e.g., a reply to a customer complaint)

Each folder can contain references to specific object types — e.g., the General info folder will be associated with an element type that only permits a link to a customer's master data in SAP R/3.

[5] The only problem with this analogy is that SAP records only contain references to objects stored elsewhere. So, to be even more accurate, envision that your medical record contains entries that remind your doctor where the actual documents are stored.

Figure 5 Example Customer Record Folders and Elements



The final term to understand is *record model*. A record model is a template that defines the structure of a record. Like a file system on a PC, the record model defines the rules that govern the creation and maintenance of records based on that model, such as where and how often each type of element can be added to a record (i.e., the element's *cardinality*). Each record derives from one and only one record model.

Now that you have a solid understanding of SAP Records Management terms and concepts, let's take SAP Records Management for a test-drive so you can see what it can do for you. First we'll set up a records management system for our example record; once we've completed this configuration, we'll create a sample record to test our setup. But before jumping right into the configuration, it's important to start with a clear picture of the record model you want to use. Otherwise, it's easy to get distracted during the configuration and lose sight of the bigger picture.

Create a Blueprint of the Record Model

To work with SAP Records Management, you will need to define a record model as a foundation for users to add items to a record, which involves defining the folder hierarchy, elements, and cardinality.

System Requirements for the Example

To be able to use SAP Records Management and set up the example, you will need SAP Web AS 6.20 or higher (with the ABAP runtime installed, or with both the ABAP and Java runtimes installed). Your SAP Web AS can underlie an application system such as SAP R/3 Enterprise (SAP R/3 4.7) or SAP CRM 4.0, or it can be a standalone SAP Web AS system. For simplicity, this article assumes you are running an SAP R/3 Enterprise system, which has the advantage that the business objects used in the example exist locally.* Also, make sure that the role SAP_BC_RM_ADMINISTRATOR is assigned to your user ID.

In addition, to be able to retrieve scanned originals, you'll need a content repository for which SAP ArchiveLink has been configured. If you haven't configured SAP ArchiveLink, you can still get this example to work by following the instructions in the document "Configuring ArchiveLink for Scanned Documents," which can be downloaded from www.SAPpro.com.

---

* If you use a standalone SAP Web AS system, you'll need to define the backend SAP application server that contains your customer records, invoices, etc. as an RFC destination. Ask your administrator how to do this. You'll need to specify the system's logical system name when defining element types.

✓ *Tip*

A record model is just a starting template for the record. End users can reorganize folders and elements, rename folders and elements, and add elements to the record beyond those specified in the record model. So, technically you can deploy a model and then give users free reign to design it. In my project experience, however, the better you define a record's structure and content up front, the more efficient the solution, and the more time savings users realize.

Before you create the actual record model in the system, it's best to sketch the model out on paper, so that you concentrate on the application and the business process and not on technical capabilities and restrictions. Ask yourself the following:

- What types of content (elements) should records contain? SAP R/3 objects? Transactions? Workflows? External documents? External data? Other records?
- How many objects of each element type can be included in the respective record (for example, *one* customer record, but *multiple* resumes)? Identify elements that are required for the record to be considered "complete."

✓ *Note!*

If you specify a cardinality of 1:1 or 1:n for an element in the record model, the element will be obligatory. The system will warn the user if links are not defined for all obligatory elements when saving, and will let the user save and flag the record as incomplete. You can write a report that calls BAPI_RECORD_GETLIST to check for incomplete records (see the system documentation if you are unfamiliar with how to do this).

- Will you need to extend or enhance the record model in the future? It's easy to subsequently add nodes to a model, but you cannot delete

nodes, since they will be required by existing records.[6]

- How can the record be structured so that the most important information can be found quickly?

> ✓ ***Tip***
>
> *Since all records based on a record model have the same structure, build your models to best accommodate the needs of all users. You can accommodate users with very different needs by defining role-based views of the model within the Records Modeler (see the sidebar on page 53).*

Figure 6 shows a sketch of our example record model. There are three things to note:

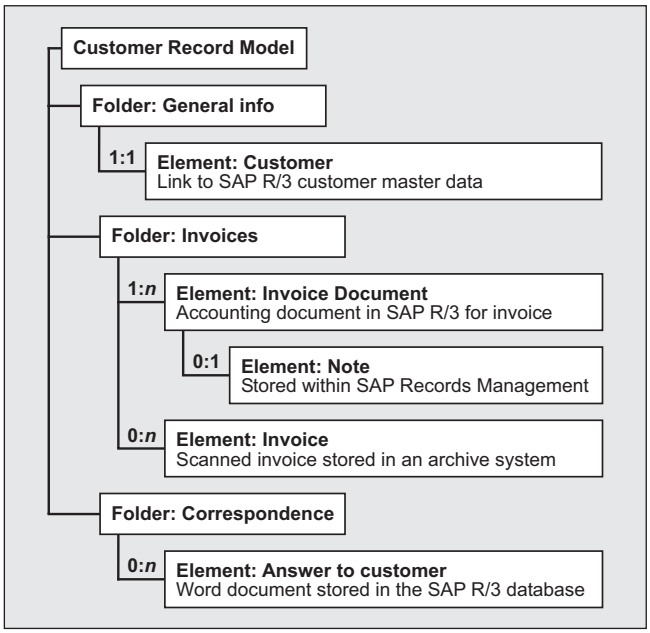
- We placed all of the elements in folders to avoid cluttering the record, and the folder names clearly suggest their content.

> ✓ ***Tip***
>
> *While there is no fixed limit to the number of folder levels and elements you can place in a record, a good rule of thumb for maximum usability and performance is to stay five or fewer levels deep and include no more than 200 elements per record. If you need to have more elements, store them in another record and include a reference to the new record in the original record.*

- Each element has a clear and logical relationship to other elements in the same record. A customer record always belongs to a customer. For every invoice there is an accounting document and perhaps additional correspondence. We also enable

[6] A node is a position in the record hierarchy. It can be a folder or an element that is included in the structure.

Figure 6 Record Model for the Example

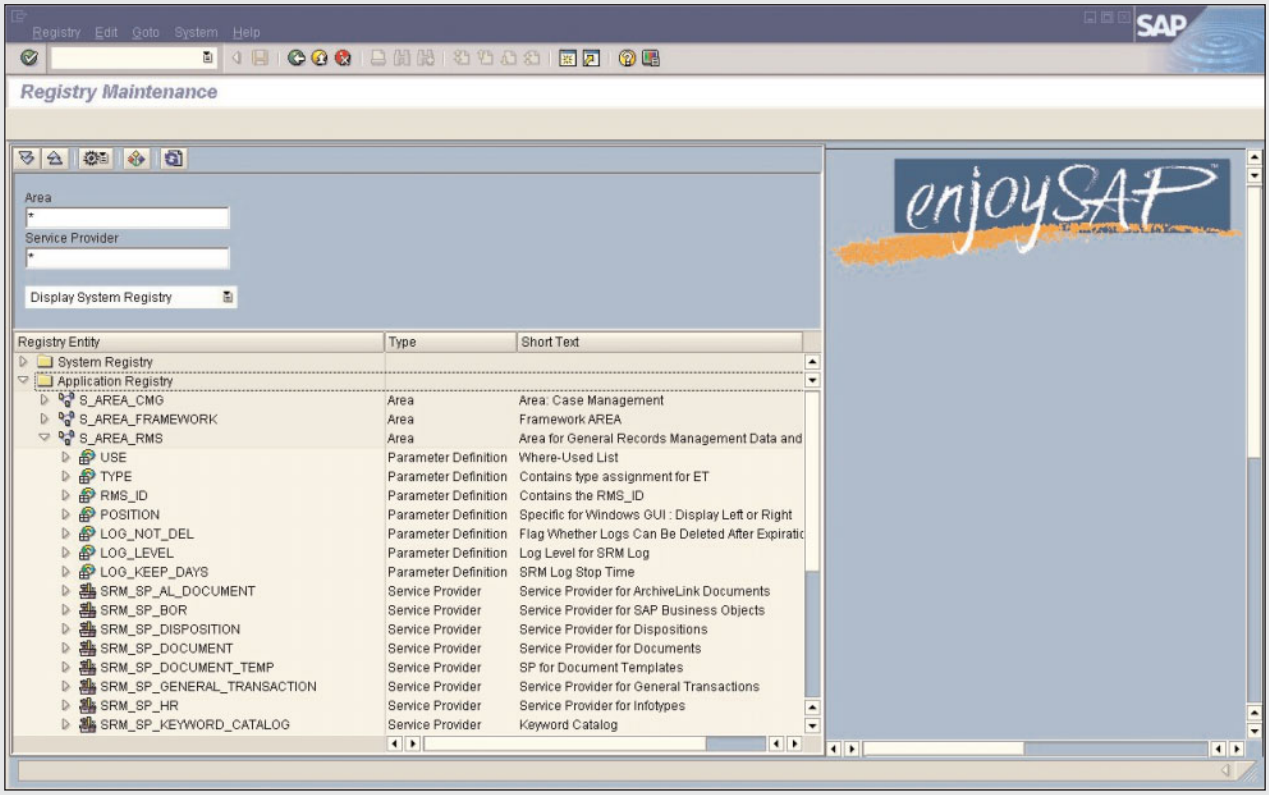


end users to create and attach notes to an invoice right from within the Records Browser, which is a standard SAP Records Management feature. Notes are managed by the note service provider, and are stored as part of the record.

- The cardinality of our example elements is as follows:
  - **Customer** — 1:1 with regard to the General info folder
  - **Invoice Document** (accounting document) — 1:*n* with regard to the Invoices folder
  - **Note** — 0:1 with regard to the Invoice Document element
  - **Invoice** (scanned invoice) — 0:*n* with regard to the Invoices folder
  - **Answer to customer** (Microsoft Word document) — 0:*n* with regard to the Correspondence folder

  Thus exactly one customer master data record and at least one invoice document (accounting document) must be associated with the record for it to be considered complete. Also, only one note

Figure 7 The Registry Lets You Maintain Parameters, Service Providers, and Element Types



can be attached to each invoice document, and an unlimited number of scanned invoices and correspondence can be added to their respective folders.

> ✓ ***Tip***
>
> *If you are unsure about which cardinality to choose, I recommend that you define it as openly as possible, using a 0:n relation.*

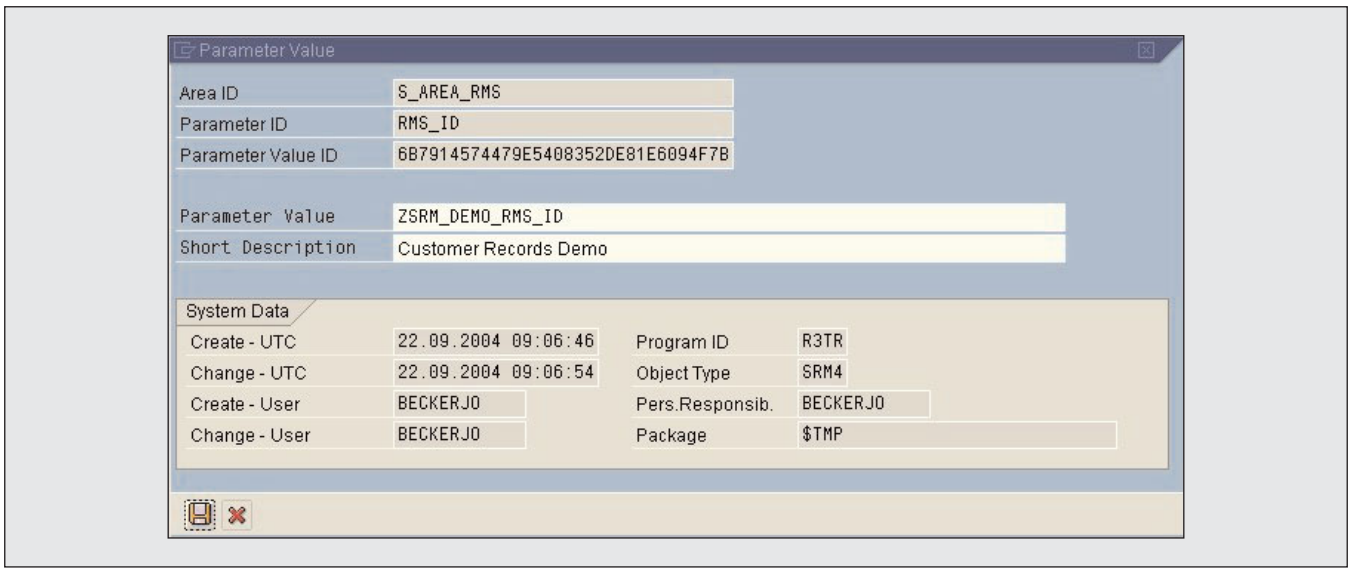
Now that we've sketched out a blueprint of our record model, we're ready to put it into action. The next section covers the configuration of our example records management system step-by-step.

Define Objects in the Registry

Underlying SAP Records Management is a tool called the Registry. The Registry lists all service providers and element types that you can use in SAP Records Management. It contains the definitions and configuration needed for applications such as SAP Records Management, including the list of records management systems you've defined for storing records. In this way, the SAP Records Management Registry functions similar to the registry in Microsoft Windows operating systems.

To define the objects for our example, launch the Registry by calling transaction SRMREGEDIT (Edit Registry). **Figure 7** displays the screen that will appear. Expand the Application Registry folder, and then expand the item S_AREA_RMS, which contains

Figure 8 Creating a New RMS_ID Parameter Value



the objects and parameters associated with SAP Records Management. (The other areas shown in Figure 7 belong to the SP Framework and are beyond the scope of this article.)

As you can see in Figure 7, S_AREA_RMS contains two types of objects: parameters and service providers. Parameters are groups of values you define that service provider applications like SAP Records Management use for classification. We will work with the RMS_ID (records management system ID) parameter, which defines all of the records management systems into which records can be stored. The service provider items represent underlying ABAP classes that access particular resources, transactions, databases, and files, and house the element types we will later need to define our example record model.

There are three things we need to do for our example:

1. Define a new records management system in the Registry for the record.

2. Determine the service providers required to access the backend content sources.

3. Identify, define, and configure the element types for the record model.

Step 1: Define a New Records Management System in the Registry for the Record

When implementing SAP Records Management, you'll want to organize your records into logically separate areas, which will allow you to control access to them as well as make them easy to navigate. You use a records management system (RMS) to do this. An RMS is similar to an SAP client — it lets you make certain groups of records accessible to only specific user groups using authorizations. Each RMS is identified by a unique RMS ID, defined as a value under the RMS_ID parameter in Figure 7.

> **Caution!**
>
> *By design, end users cannot search across RMSs, so try to keep related records in a single RMS. Divide unrelated records into separate RMSs, to keep record volumes reasonable and help reduce the volume of search results.*

To show you how to set up your own RMS, let's define one for our example. Right-click on the RMS_ID parameter in the registry tree shown in Figure 7 and choose Create Parameter Value from the context menu. On the dialog that appears (**Figure 8**), enter ZSRM_DEMO_RMS_ID as the technical name, Customer Records Demo as the description, and save. The system automatically generates a unique, cryptic parameter ID for the value, which you can ignore.

Before continuing, verify that a new RMS has been created by expanding the RMS_ID item previously shown in Figure 7.

Step 2: Determine the Service Providers Required to Access the Backend Content Sources

As I mentioned previously, service providers consist of ABAP classes that handle the specifics of accessing backend resources. Each service provider is responsible for a certain number of information objects (see the sidebar below).

Fortunately, all of the service providers we need for our example already exist in the SAP Web AS system, so we do not have to do any programming.

Standard SAP-Delivered Service Providers for SAP Records Management

The table below lists the standard service providers included in SAP Web AS 6.20 and higher, and what they are used for.

Service Provider	Use
SRM_SP_BOR	Business objects
SRM_SP_HR	HR infotypes
SRM_SP_GENERAL_TRANSACTION	General transactions
SRM_SP_REPORT	SAP system reports
SRM_SP_AL_DOCUMENT	ArchiveLink documents
SRM_SP_NOTE	Simple notes
SRM_SP_DOCUMENT	Documents managed by the Knowledge Provider,* including document versions and document attributes
SRM_SP_PAPERDOCUMENT	Paper documents (i.e., metadata for a document and a description of where to find the original)
SRM_SP_DOCUMENT_TEMP	Document templates
SRM_SP_URL	URLs pointing to the Internet or intranet
SRM_SP_KEYWORD_CATALOG	Keyword catalog for searching
SRM_SP_RECORD	Records
SRM_SP_MODEL	Record models
SRM_SP_PLAN	Storage plans for records
SRM_SP_REFERENCE	Record number generator
SRM_SP_WORKFLOW	Business workflows
SRM_SP_DISPOSITION	Disposition or circular (document-based approval process)

* The Knowledge Provider is a generic document management infrastructure included with SAP Web AS.

The service providers we'll use are:

- **SRM_SP_BOR**, which provides access to business objects in an SAP system. In our example, we will use this service provider to access customer master records via the Customer business object, defined in the Business Object Repository (BOR), as well as accounting documents in the SAP R/3 system.[7] At runtime, the service provider calls methods on the business object in the target SAP application system.
- **SRM_SP_AL_DOCUMENT**, which provides access to scanned originals on an external content server (an archive system or the standard SAP Web AS Content Server) configured for SAP ArchiveLink.[8] We will use this service provider to enable access to scanned invoices.
- **SRM_SP_NOTE**, which lets users create and maintain notes. In our example, we will use this service provider to enable users to attach notes to accounting documents via the SAP Records Management notes functionality.
- **SRM_SP_DOCUMENT**, which provides access to documents that are stored on any content server that can be accessed by the Knowledge Provider built into all SAP Web AS systems[9] — e.g., the SAP Web AS Content Server or an external optical archive. We will use this service provider to enable correspondence to be created and edited in Microsoft Word (refer back to Figure 2 for an illustration of this capability) and saved to the SAP R/3 database.
- **SRM_SP_RECORD**, which provides methods to create, modify, and access records.
- **SRM_SP_MODEL**, which provides methods to work with record models.

> ✓ ***Remember!***
>
> *In many cases, you'll want to develop your own service providers to access non-SAP systems like SQL databases or other third-party systems. For more information on how to do this, visit the documentation area at http://service.sap.com/recordsmanagement.*

Step 3: Identify, Define, and Configure the Element Types for the Record Model

In the Registry, element types are associated with the service providers that let SAP Records Management access their content — e.g., the element type for the Customer business object will be placed under the service provider for business objects. One or more element types can be configured for each service provider. Element types give the service provider more details on the information objects the service provider will access, such as the name and target system of the BOR object that the service provider for business objects should call in order to access the desired customer master data. The element types you have defined for a service provider can be displayed by expanding the node beneath the service provider in the Registry (Figure 7).

Defining element types involves three steps:

1. Create the element type by right-clicking on the associated service provider and choosing Create Element Type from the context menu.
2. Assign values to the connection parameters, so the service provider can connect to the appropriate data source.
3. Assign classification values to the objects, to make sure that the element types appear in the correct RMS, for example.

---

[7] The Business Object Repository (transaction SWO1) is a centralized registry of the business object types in your SAP system, such as Customer, Sales Order, etc. It documents the attributes for each object type, and the methods (functions) that can be called to create, display, or change business objects of that type.

[8] Remember that if you haven't configured SAP ArchiveLink, you can still get this example to work by following the instructions in the download available from www.SAPpro.com.

[9] The Knowledge Provider is a document management infrastructure included as part of SAP Web AS that supports metadata definition, versioning, multilanguage capabilities, format independence, and the actual storage location of digital documents.

For our example, we need to define the following element types:

- **Customer**, associated with the service provider for business objects (SRM_SP_BOR)
- **Accounting Document**, also associated with the service provider for business objects
- **Note**, associated with the service provider for notes (SRM_SP_NOTE)
- **Scanned Invoice**, associated with the service provider for ArchiveLink documents (SRM_SP_AL_DOCUMENT)
- **Correspondence**, associated with the service provider for documents (SRM_SP_DOCUMENT)

We also need to define element types for the record we will create (associated with service provider SRM_SP_RECORD) and the record model we will define (associated with service provider SRM_SP_MODEL). This lets records contain or reference each other, for example. The element types we need to define, and their corresponding values, are summarized in **Figure 9**. I will describe where to place each of these values next.

To gain some experience, let's create an element type for the Customer element. As shown in Figure 9, access to a Customer object (a business object in nearly all SAP application systems) is facilitated by the SRM_SP_BOR service provider, so start by right-clicking on SRM_SP_BOR in the Registry (Figure 7) and choosing Create Element Type.

Next, enter a technical name and short description for your element type — for example, ZSRM_SPS_BO_CUSTOMER and Customer, respectively — and save the new element type. In the transport dialog, press the Local Object button to save it as a local object.

Figure 9 List of Element Types Required for the Example

Element (Description)	Element Type (Technical Name)	Service Provider	Connection Parameters	Classification
Customer (customer master data in the SAP R/3 system)	Customer (ZSRM_SPS_BO_ CUSTOMER)	SRM_SP_BOR	BOR_OBJECT_TYPE = KNA1; LOGICAL_SYSTEM = NONE;*	TYPE = SRM_BUSINESSOBJECT RMS_ID = Z_DEMO_RMS_ID
Invoice Document (accounting document posted in FI in the SAP R/3 system)	Accounting Document (ZSRM_SPS_BO_ ACC_DOC)	SRM_SP_BOR	BOR_OBJECT_TYPE = BKPF; LOGICAL_SYSTEM = NONE;* METHOD_BOR_ OBJECT_DISPLAY = DISPLAY	TYPE = SRM_BUSINESSOBJECT (or SRM_DOCUMENT**) RMS_ID = Z_DEMO_RMS_ID
Note (can be attached to accounting documents)	Note (ZSRM_SPS_NOTE)	SRM_SP_NOTE	DOCUMENT_CLASS = SRM_DOC04	TYPE = SRM_DOCUMENT RMS_ID = Z_DEMO_RMS_ID
Invoice (scanned originals created with SAP ArchiveLink)	Scanned Invoice (ZSRM_SPS_AL_ INVOICE)	SRM_SP_AL_ DOCUMENT	BO_TYPE = BKPF; CREP_ID = <your content repository>	TYPE = SRM_DOCUMENT RMS_ID = Z_DEMO_RMS_ID
Answer to customer (document created and edited in Microsoft Word and stored in the SAP R/3 system)	Correspondence (ZSRM_SPS_ OFFICE_DOC)	SRM_SP_ DOCUMENT	DOCUMENT_CLASS = SRM_DOC04	TYPE = SRM_DOCUMENT RMS_ID = Z_DEMO_RMS_ID

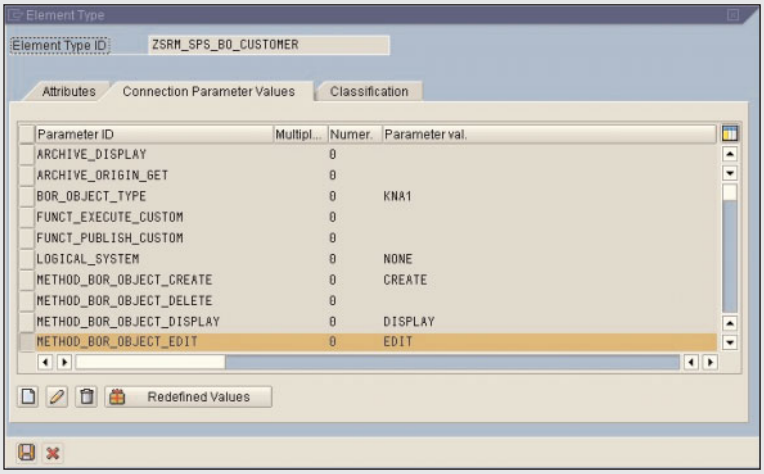
(continued on next page)

Figure 9 (continued)

Element (Description)	Element Type (Technical Name)	Service Provider	Connection Parameters	Classification
Customer Record (container for the record elements)	Customer Record (ZSRM_SPS_ CUSTOMER_ RECORD)	SRM_SP_ RECORD	DOCUMENT_CLASS = SRM_REC00; MODEL_ID = <your record model>	TYPE = SRM_RECORD RMS_ID = Z_DEMO_RMS_ID
Customer Record Model (definition of the record structure)	Demo Model (ZSRM_SPS_ DEMO_MODEL)	SRM_SP_ MODEL	DOCUMENT_CLASS = SRM_MOD02	TYPE = SRM_RECORDMODEL RMS_ID = Z_DEMO_RMS_ID

* If you are using a standalone SAP Web AS for the example, enter the logical system name of your backend SAP application system instead.
** If you choose SRM_DOCUMENT, this element will appear in the Documents folder in the Records Organizer instead of in the Business Objects folder.

Figure 10 Connection Parameter Values for the Example



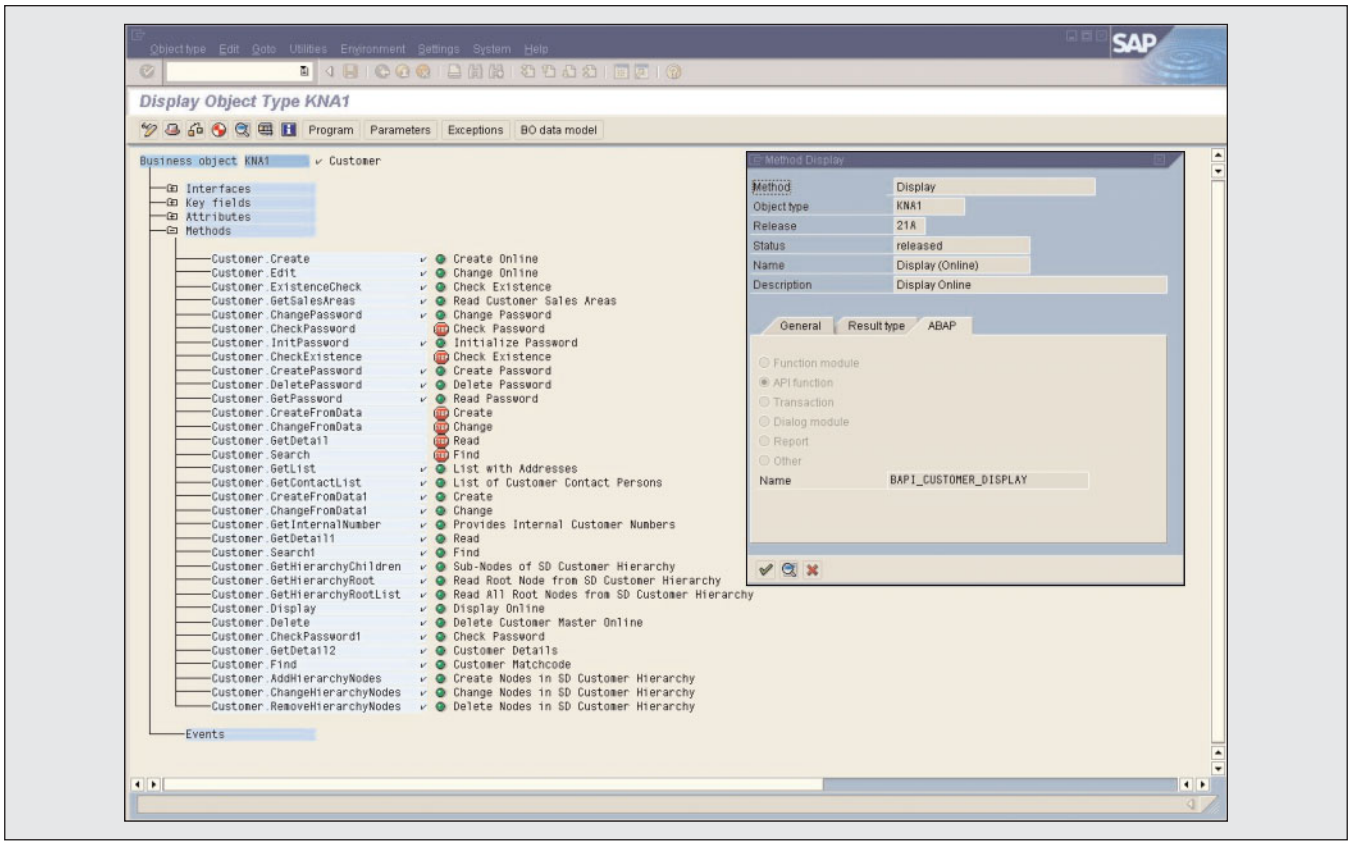
> ✓ ***Note!***
>
> *Don't exit the element type dialog just yet — we are not done. If you do, re-launch it by right-clicking on the element type and choosing Change from the context menu.*

Next, we need to assign values to the connection parameters. Navigate to the Connection Parameter Values tab (see **Figure 10**), and specify the following:

- **KNA1** for the BOR_OBJECT_TYPE parameter
- **NONE** for the LOGICAL_SYSTEM parameter
- **DISPLAY** for the METHOD_BOR_OBJECT_DISPLAY parameter

> ***Caution!***
>
> *Always enter the parameter values in uppercase to ensure that the system interprets them correctly.*

Figure 11 Methods of the Customer Object in the BOR (Transaction SWO1)

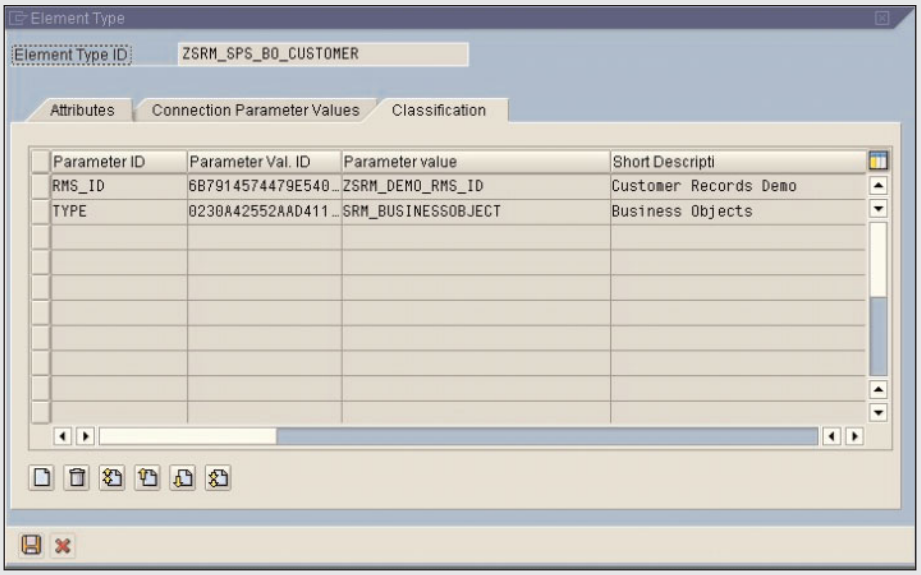


> ✓ ***Note!***
>
> *If you also want to create or change customers from within SAP Records Management, just fill in the METHOD_BOR_OBJECT_CREATE or METHOD_BOR_OBJECT_EDIT parameters with the values CREATE and EDIT, respectively, as shown in Figure 10.*

If you are using a standalone SAP Web AS system, enter the logical system name of your backend SAP application system in the LOGICAL_SYSTEM field. The specific connection parameters you'll see depends on the service provider the element type belongs to. Parameters for standard SAP service providers are thoroughly documented in the online help for SAP Records Management at http://help.sap.com. Here, I'll only describe the parameters needed for the example.

The first parameter (BOR_OBJECT_TYPE) links our new Customer element type to business object type KNA1 in the Business Object Repository. The second parameter (LOGICAL_SYSTEM) tells the service provider that the Customer's master data is located in the local system (versus a remote SAP system; remember that the example assumes an SAP R/3 Enterprise system with an underlying SAP Web AS rather than a standalone SAP Web AS). The third parameter (METHOD_BOR_OBJECT_DISPLAY) tells the service provider which method to call when the user tries to drill down on this element in the Records Browser. Methods in the Business Object Repository are usually linked to ABAP function modules (BAPIs) or ABAP classes that implement their functionality. **Figure 11** shows the methods for the

Figure 12 Classifying the Example Customer Element Type



Customer object displayed in the Business Object Repository (transaction SWO1).

> ✓ ***Note!***
>
> *When entering the connection parameters, the system may prompt you to select an RMS ID. Choose the RMS ID we created earlier (ZSRM_DEMO_RMS_ID).*

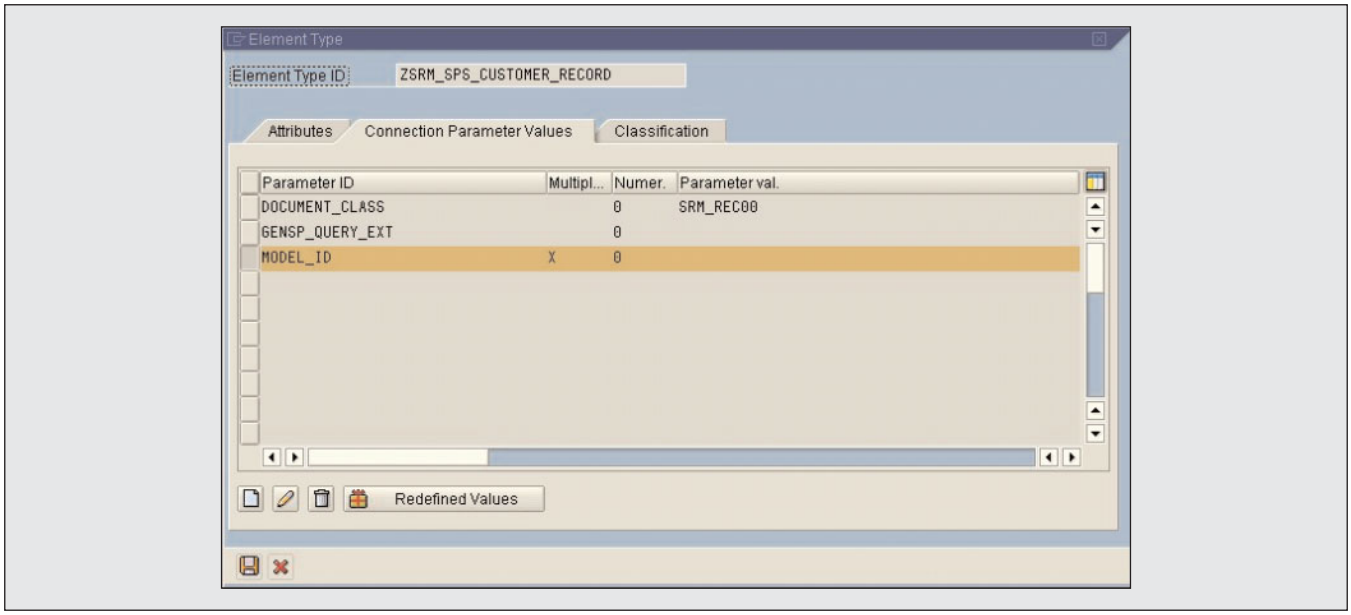
The final thing we need to do is classify our new element type by specifying parameter values on the Classification tab (see **Figure 12**). These parameters help SAP Records Management organize the large number of element types in a productive system.

For the purposes of the example, we will use just two classification parameters:

- The **RMS_ID** parameter identifies the RMS ID to which the element type is bound, as described earlier. Without this value, the element type won't appear in the Records Organizer (more on this in the next section) or in the element type selection dialog we'll use when defining our record model. Since we want to assign all of our element types to our new RMS ID, specify ZSRM_DEMO_RMS_ID for all of the element types you create.
- The **TYPE** parameter classifies the element type according to a standard set of types shipped with SAP Web AS. The value you need to specify depends on the type of element that you are creating. The standard types include business objects, documents (for notes, correspondence, scanned documents, etc.), records, and record models, plus a few others we don't need for the example. As indicated earlier in Figure 9, specify SRM_BUSINESSOBJECT for our Customer element type.

Just to see how the values differ, let's quickly create the element type for our example customer record. To do this, select the SRM_SP_RECORD service provider and create the new element type ZSRM_SPS_CUSTOMER_RECORD with the short description Customer Record. Then, navigate to the Connection Parameter Values tab as before (see **Figure 13**).

Figure 13 Connection Parameters for the Customer Record Element Type



Notice in Figure 13 how the parameters proposed by the system for the Customer Record element type differ from those for our Customer element type. This is because each data source has a unique interface, and shows how the SP Framework accommodates these differences in a standard way. In this case, the service provider for records asks for two values of importance[10]:

- **DOCUMENT_CLASS** defines the repository where the records are stored, as well as the attributes you can maintain for the record. As shown in Figure 13, enter SRM_REC00 for the DOCUMENT_CLASS, which is a standard-delivered class.
- **MODEL_ID** specifies the record model the system should use for records with a record element type. Since we have not yet created our record model, leave this parameter undefined and save.

Finally, switch to the Classification tab (Figure 12) and enter ZSRM_DEMO_RMS_ID for RMS_ID and SRM_RECORD for TYPE.

[10] The parameter GENSP_QUERY_EXT is used for special search queries, and is not relevant here.

With these examples, creating the remaining element types should be straightforward using the values listed in Figure 9.

Define the Record Model

Now that we've defined our RMS ID, determined the necessary service providers, and defined the element types we'll use, we can build the record model that associates them. This involves four steps:

1. Launch the Records Organizer.
2. Create a new record model based on the defined record model element type.
3. Add nodes to the record model, using the element types defined in the previous section.
4. Activate and link the new record model.

Step 1: Launch the Records Organizer

Like all SAP Records Management tools, the Records Modeler is accessed via the Records Organizer,

Figure 14 SAP Records Management Initial Screen — Records Organizer

> ✓ ***Note!***
>
> *To start the Records Organizer, you need the permissions of the authorization object S_SRMSY_CL. This is included in the SAP role SAP_BC_RM_USER (the standard user for SAP Records Management).*

transaction ORGANIZER (see **Figure 14**). The Records Organizer acts as a central desktop from which you can create and process records, documents, models, etc. It automatically launches the Records Browser and Records Modeler tools as needed. It is also a tool that end users can use to view and maintain records, as shown back in Figure 1 (see the sidebar on the next page for more on the end user perspective of the Records Organizer).

When you call the Records Organizer for the first time, the system will ask you to enter a records management system ID (RMS ID). For the example, enter ZSRM_DEMO_RMS_ID, which is the ID

End Users and the Records Organizer

The Records Organizer is more than just a configuration tool — it is also a central launchpad for end users to locate, display, and modify records in SAP Records Management. Figure 14 shows the standard view with all of the configuration options. Usually, however, you'll want to develop role-based views that contain only the options a user needs — e.g., you might want to hide the record model item from the tree to avoid confusing users with extra options. You create role-based views for the Records Organizer in the IMG — the path is *SAP Web Application Server → Basis Services → Records Management → Create Role-Based View*.

In addition to, or in lieu of, the Records Organizer, many companies provide users with direct links into SAP Records Management from their SAP applications. For example you could add a Display Record menu item to the Object Services menu of the Display Customer Master transaction, and configure that menu item to launch the associated customer record in the Records Browser.* The menu item would call a BAPI like BAPI_RECORD_DISPLAY, which is associated with the display method of the record business object (see the system documentation for more details).

* For more on adding links to the configurable generic Object Services menu (available in many SAP transactions), go to the SAP Web AS 6.20 documentation at http://help.sap.com and navigate to *mySAP Technology Components → SAP Web Application Server → Basis Services and Communication Interfaces → Business Workplace and Services → Generic Object Services*.

we created earlier. The Change RMS button ( ) located in the Records Organizer can be used to switch to another RMS, provided you have the required authorizations.

The upper section of the navigation area shown in Figure 14 offers three views: Role-Based View, Favorites, and Resubmission.[11] You can switch between these views by choosing the appropriate button at the top of the display area. The lower section contains a History view providing easy access to elements you have recently accessed.

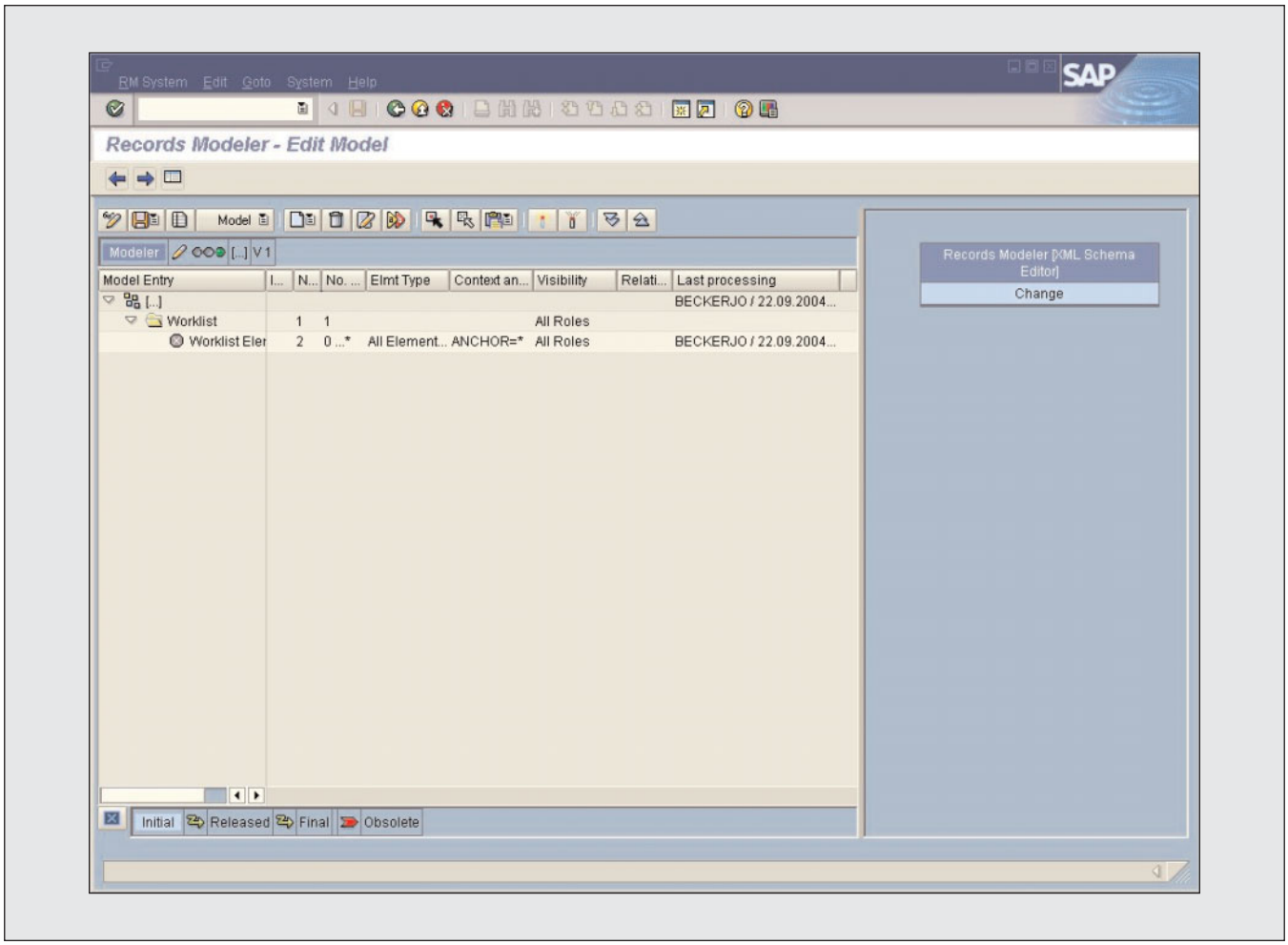
The role-based view lets you check new elements into SAP Records Management, or search for elements that have already been checked in. As delivered, the role-based view consists of three hierarchical levels. The first contains only one node representing the records management system you entered when launching the Records Organizer. The second level contains folders that group together elements under different headings, like Business Objects, Records, Record Models, and Documents. Element types are grouped by the classification you specified in the Registry. The third level contains the actual element types. Before continuing, verify that each of the element types you defined earlier appear in their respective folders. As you can see, all of the element types in Figure 9 appear in the folders in Figure 14.

[11] We will only cover the standard role-based view in this article. The favorites view lets you save links to certain records, documents, or document templates (forms) you use frequently, just like in your Internet browser. The resubmission view contains reminders you can set about certain records and documents — e.g., a reminder to reexamine the budget plan as soon as more information is available.

✓ ***Note!***

You will only see those element types that have been classified for the corresponding RMS_ID and type. If one you defined earlier doesn't appear, go back and verify that its RMS_ID and type classification parameter values have been set appropriately.

Figure 15 *The Records Modeler*



Step 2: Create a New Record Model Based on the Defined Record Model Element Type

To define our new record model, right-click on the Demo Model element type in the Record Models subdirectory, and select Create from the context menu. The Records Modeler will appear (see **Figure 15**).

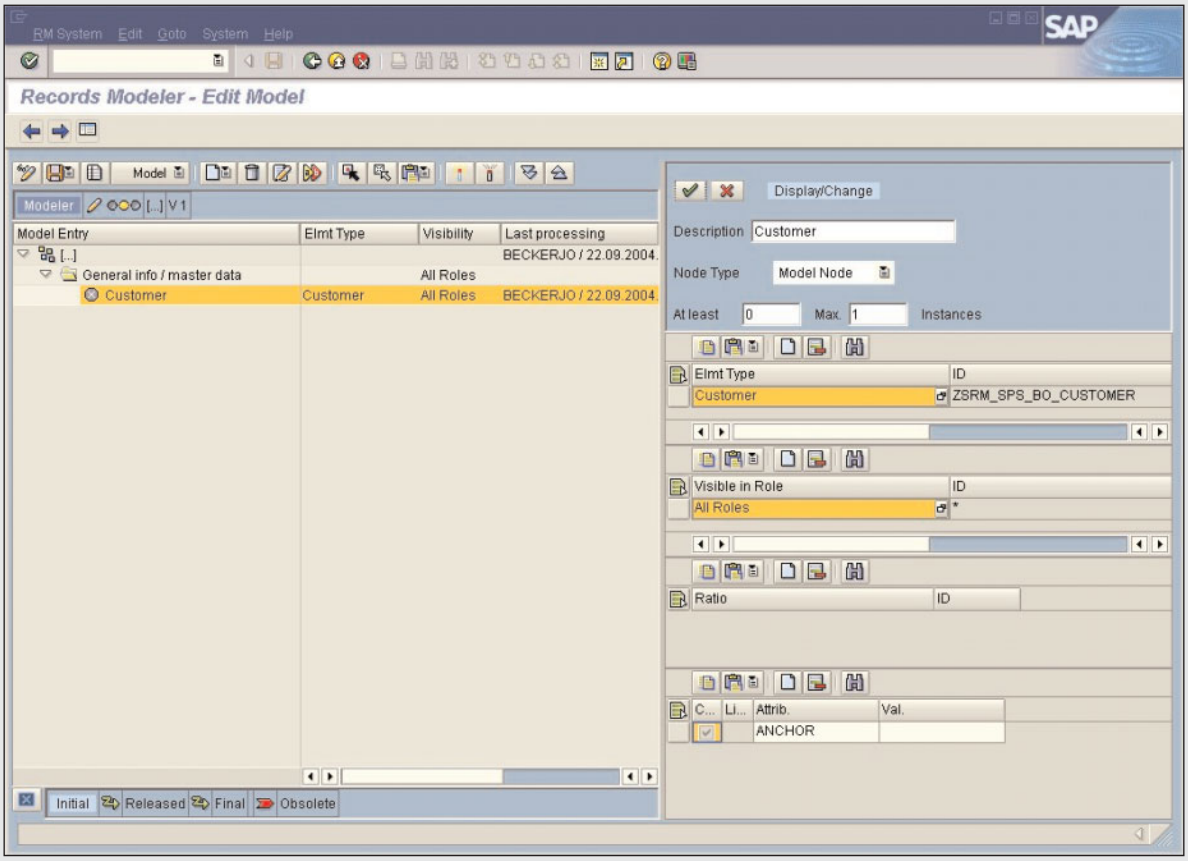
The base of our new record model appears at the top of the navigation pane ([…] in Figure 15). Below the record model are nodes. If you expand the overview tree, the default setting displays a header "structure" node (Worklist in Figure 15) and a below it a "model" node (Worklist Element in Figure 15).

Step 3: Add Nodes to the Record Model

Adding a new node is very easy. Simply right-click on the node before or after which you want to insert a new node, and use one of the functions in the context menu ("Create on same level before" or "Create on same level after"). The detailed display of the new node will appear in the screen area on the right (see **Figure 16**), where you maintain a description, the node type, and the cardinality (at least 0 and a maximum of 1 instance).

While creating nodes is clearly very simple, there are two settings in Figure 16 that warrant further discussion: node types and visibility. First, as you can see in Figure 16, the node's type is selected

Figure 16 *Defining a New Node*



from a dropdown box. There are three node types to choose from:

- **Structure nodes**, like General info, appear as folders, and cannot have elements directly assigned to them. These nodes only act as headers for other structure nodes and for model nodes. You can include an unlimited number of structure nodes in your record model.

- **Model nodes**, like Customer, appear with gray placeholder icons in new records, so users can clearly identify which nodes are empty. You can include an unlimited number of model nodes in your record model. The element type you assign to a model node restricts the types of elements that can be assigned to that node — e.g., an SAP R/3 accounting document to the Invoice Document node or a Microsoft Office document to the Answer to customer node. In the example, only customer master data can be assigned to the Customer node.[12]

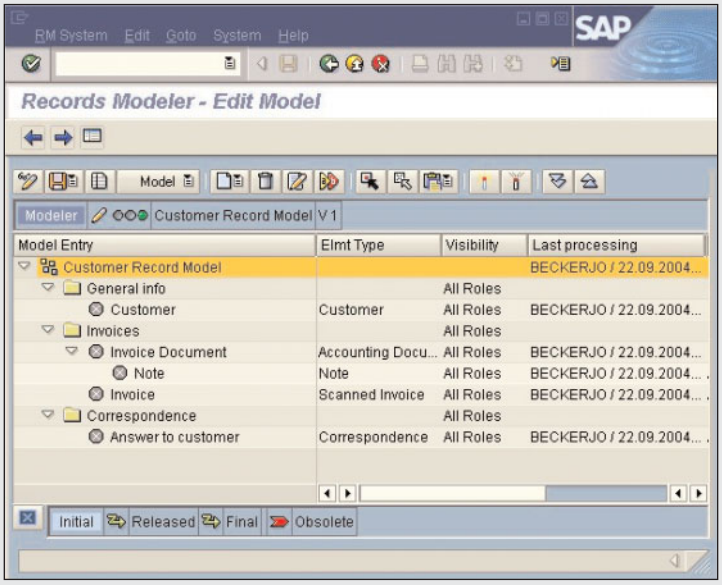
- **Instance nodes** are infrequently used. You use them if you want a certain information object in all records based on this model — e.g., a help document telling employees what to include in the record. There can only be one instance node in a record.

[12] The ratio and attribute settings in Figure 16 are optional and not relevant to the discussion here.

Figure 17 Nodes to Add to the Example Record Model

Node Name/Description	Node Type	Element Type	Visibility	Cardinality
General info	Structure node	N/A	All Roles	—
Customer	Model node	Customer	All Roles	1:1
Invoices	Structure node	N/A	All Roles	—
Invoice Document	Model node	Accounting Document	All Roles	1:n
Note	Model node	Note	All Roles	0:1
Invoice	Model node	Scanned Invoice	All Roles	0:n
Correspondence	Structure node	N/A	All Roles	—
Answer to customer	Model node	Correspondence	All Roles	0:n

Figure 18 Finished Record Model for the Example (Not Yet Released)



For the example, we require only structure nodes and model nodes.

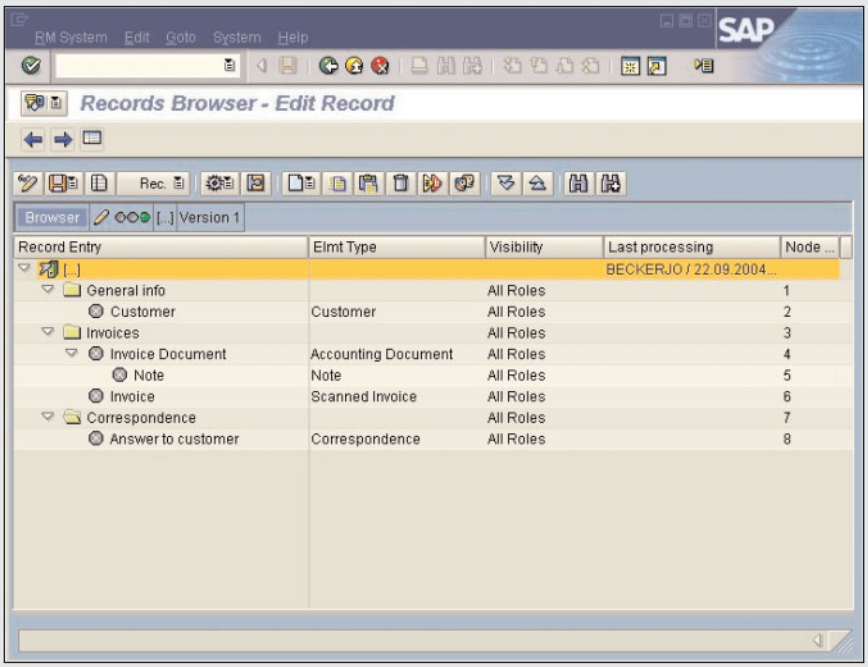
The second field of importance is the Visible in Role field. You can use this field to narrow the visibility of a specific node to one or more SAP R/3 user roles (which are defined as a standard part of SAP security with transaction PFCG). For simplicity, in the example we'll specify All Roles.

Figure 17 lists the nodes we need to create, and the values to use for each. When you are finished, click on the green checkmark to apply the settings. The finished record model should look like **Figure 18**.

Step 4: Activate and Link the New Record Model

Before you can start creating records, you need to release and save the record model. To do this, choose *Activity → Model → Change Status* from the menu bar, then select Released status. This releases the

Figure 19 Example Record in Initial State (Empty)



> ***Caution!***
>
> *Now that you have created and released your record model, you must go back to the Registry and enter a value for the connection parameter MODEL_ID for your record element type ZSRM_SPS_CUSTOMER_RECORD (use input help to enter the record model that you created). This tells the system that any future customer records are based on this record model. If you forget, end users will be forced to enter a record model before they can work with the system. To enter the value, launch the Registry, right-click on the record element type ZSRM_SPS_CUSTOMER_RECORD, select edit, and enter the value on the Connection Parameter Values tab as described earlier (refer back to Figure 13).*

record model for general use. Finally, click on the Save toolbar button and enter Customer Record Model for both the Short Description and Unique ID attributes in the dialog box that appears. The Unique ID attribute provides the system with a unique identifier for the model, even if the description is identical to other models.

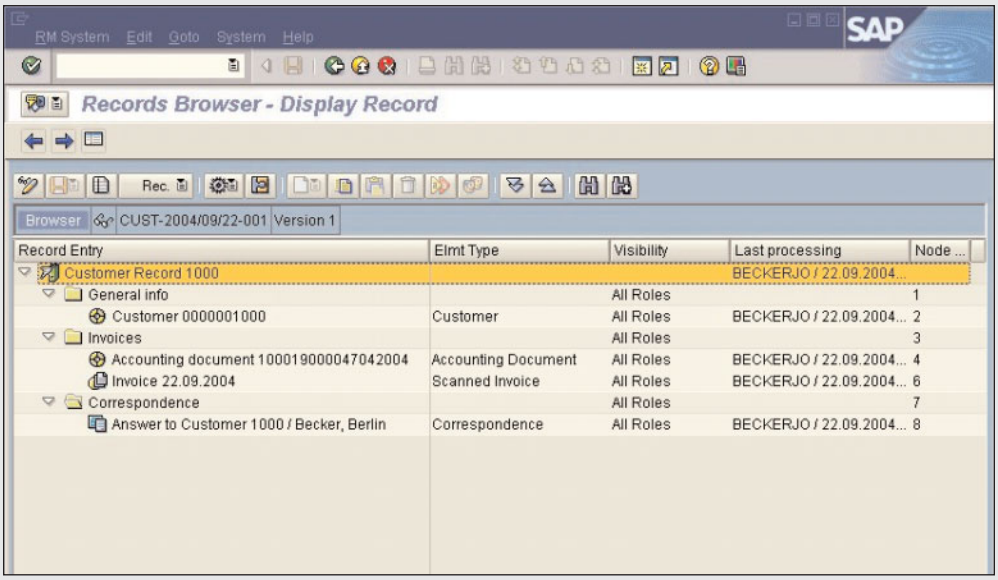
We're now ready to create a sample record based on our new model.

Adding a Record to the Example RMS

To test our configuration, let's create the example customer record introduced in Figure 5. First, switch back to the Records Organizer (which is still active in the background). If you are still in the Records Modeler, click on the icon to the right of the blue arrows ( ).

Next, right-click on the Customer Record element type, and choose Create from the context menu (refer back to Figure 14). The Records Browser will appear as shown in **Figure 19**, with the structure we defined in our record model.

Figure 20 Example Record Populated by Object References



Add references to a customer master record, invoice accounting document, and scanned invoice by right-clicking on the appropriate node and selecting Find or Create from the context menu. Choose Find to search for existing objects in the underlying system, or Create to create new objects — e.g., a new invoice or customer in SAP R/3. In our example, we defined the Customer and Invoice Document elements as obligatory (i.e., a cardinality beginning with 1), so the system will warn you that the record is incomplete if you try to save without assigning a reference to these elements.

> ✓ ***Note!***
>
> *When you choose Create for the "Answer to customer" element, the system will ask you to select the Microsoft Office application you want to use (in this case, Word).*

> ✓ ***Note!***
>
> *Alternatively, you can upload an existing local file by choosing Create from File instead.*

A fully populated record will look like the one in **Figure 20**.

We have now created an example record based on our example customer record model. To gain further experience, create several records for different customers, experiment with restructuring and renaming items, and try searching for records within the Records Organizer. This will give you a sense of what the system can do, which functions you will want to roll out to users, and which functions you will want to restrict.

Tips for Implementing SAP Records Management

Here are some things to remember when implementing and using SAP Records Management, and some potential enhancements you'll want to consider as a part of your rollout:

☑ **Record models can be defined generically or specifically.** In the example, we designed our record model to exclusively accommodate customer records. We could have defined the model more generically, so it could be used for *any* type of business partner. First, we would not use any type-specific words like "Customer" in our folder and element labels. Second, we would make sure all business object references were type-neutral — i.e., we would have used the more generic Business Partner business object (found in nearly all SAP systems) instead of the Customer business object. Since Customer and Vendor are subtypes of the Business Partner object, the link to the business partner could then contain a reference to either a customer or vendor master record.[13] In general, however, it's best to define specific models to improve usability. Nevertheless, this can be a great way to reduce the number of record models you need to maintain when a type-specific design isn't critical.

☑ **Develop additional service providers to expand your content options.** The majority of projects don't need to implement new service providers; however, you may want to implement a new service provider if you store records in Lotus Notes, SQL databases, or other third-party systems. For details on how to do this, as well as additional tutorials and information, see the documentation area at http://service.sap.com/recordsmanagement.

☑ **Leverage workflow to automatically create and populate records.** In the example, we populated our first record manually. For maximum efficiency, you'll probably want to set up your system to create and populate records automatically as other objects are created. Technically, this is not hard to do, but it involves some planning: the SAP Records Management team added a set of BAPIs to the RECORD business object type that you can call from custom ABAP programs, user exits, or workflows (see transaction SWO1).[14] So, if you want to automatically create a new customer record every time a new customer is created with transaction KNA1, for example, bind a new workflow to the CUSTOMER.CREATE event that calls BAPI_RECORD_CREATE.

> ✓ ***Note!***
>
> *A more efficient approach would be to write a custom ABAP event receiver function module and bind it directly to the CUSTOMER.CREATE event, rather than use a workflow. You can define the event linkage by following the SAP standard menu path Tools → Business Workflow → Development → Administration → Event manager → Type linkages.*

After creating the record, you can add a link to the customer by creating an element of the record using BAPI_RECORD_ADDELEMENT — we did this manually in our example. Once you've got this working, define new workflows (or modify existing ones) to populate your record with additional links — e.g., new invoice scans in the archiving system.

☑ **SAP Records Management can be accessed from the Web.** This is particularly useful if your users use SAP Enterprise Portal as their central workplace. In the Internet Connection Manager (transaction code SICF), activate the Business Server Pages (BSP) application SRMCLFRM and/or enter an alias for the URL. Next, define a

[13] You can investigate class relationships in the Business Object Repository (transaction SWO1).

[14] The BAPIs are registered as methods of business object type RECORD in the Business Object Repository (transaction SWO1), and are documented in detail.

role-based view of your record model and assign it to your user ID (follow the IMG menu path *SAP Web Application Server → Basis Services → Records Management → Create Role-Based View*). You can then access SAP Records Management via the URL http://<Default-Host>/<alias>/start.htm. For more instructions, go to the online help at http://help.sap.com and navigate to *SAP NetWeaver Components → SAP Records Management → Customizing → Web Display Customizing*.

Conclusion

SAP Records Management offers a wealth of opportunities for SAP customers to integrate transactional data, documents, workflows, and other data from SAP and non-SAP systems via a new, easily extensible framework called the Service Provider Framework. SAP Records Management is included as a part of SAP Web Application Server 6.20 and higher (ABAP or ABAP+Java version), but is licensed separately.

This article provided a first look at what SAP Records Management can do, and how to begin constructing custom record models. There's much more functionality to explore, however, including:

- Managing business processes with SAP Records Management using workflow, circulars, and cases
- Defining attributes for digital records via "content models" to more easily organize and link to your records
- Using external content servers and alternatives for secure storage of digital documents
- Leveraging document templates to simplify creating and editing documents
- Using the document finder for more efficient document retrieval
- Performing full text searches and indexing documents
- Creating role-based views
- Exploring Case Management, which is based on SAP Records Management and used in SAP CRM and in mySAP Financials

Try exploring these various functionalities, and see what SAP Records Management can do for you. You can find more information on these topics by visiting the SAP Records Management Web page in the SAP Service Marketplace at http://service.sap.com/recordsmanagement.

Joachim Becker is Product Manager of SAP Records Management. He has a degree in mathematics from the University of Heidelberg, Germany. In his degree dissertation, Joachim discussed wavelet algorithms, which are used to analyze digital images, for example. He joined SAP in 1995 as a developer in the Financials area, and also worked as a trainer and consultant. Since 1999, Joachim has assumed various product management responsibilities in SAP's Technology Development department. He is an expert in Business Process Technology in general, and has focused on the areas of Workflow, Document Management, and Communication. Joachim may be contacted at joachim.becker@sap.com.

