In this selection from Chapter 5, you’ll get a taste of some of the main building blocks that make the jobs run in SAP Data Services. This chapter provides both the information you need to understand each of the objects, as well as helpful step-by-step instructions to get them set up and running in your system.
Introduction

Welcome to SAP Data Services: The Comprehensive Guide. The mission of this book is to capture information in one source that will show you how to plan, develop, implement, and perfect SAP Data Services jobs to perform data-provisioning processes simply, quickly, and accurately.

This book is intended for those who have years of experience with SAP Data Services (which we’ll mainly refer to as just Data Services) and its predecessor, Data Integrator, as well as those who are brand new to this toolset. The book is designed to be useful for architects, developers, data stewards, IT operations, data acquisition and BI teams, and management teams.

Structure of the Book

This book is divided into four primary sections.

- **PART I: Installation and Configuration (Chapters 1–3)**
  We start by looking at planning out your systems, the installation process, how the system operates, and how to support and maintain your environments.

- **PART II: Jobs in SAP Data Services (Chapters 4–8)**
  The book then delves into the functionality provided in Data Services and how to leverage that functionality to build data-provisioning processes. You are taken through the process of using the Integrated Development Environments, which are the building blocks for creating data-provisioning processes.

- **PART III: Applied Integrations and Design Considerations (Chapters 9–12)**
  Here you’re taken through use cases that leverage Data Services, from requirements to developed solution. Additionally, design considerations are detailed to make sure the data-provisioning processes that you created perform well and are efficient long after they are released into production environments.

- **PART IV: Special Topics (Chapters 13–14)**
  The book then explores how SAP Information Steward, when used in conjunction with Data Services, enables transparency of data-provisioning processes.
and other key functionality for organizations’ EIM strategy and solutions. Finally, the book looks into the outlook of Data Services.

With these four parts, the aim is to give you a navigation panel on where you should start reading first. Thus, if your focus is more operational in nature, you’ll likely focus on Part I, and if you’re a developer new to Data Services, you’ll want to focus on Parts II and III.

The following is a detailed description of each chapter.

**Chapter 1. System Considerations**, leads technical infrastructure resources, developers, and IT managers through planning and implementation of the Data Services environment. The chapter starts with a look at identifying the requirements of the Data Services environments for your organization over a five-year horizon and how to size the environments appropriately.

**Chapter 2. Installation**, is for technical personnel. We explain the process of installing Data Services on Windows and Linux environments.

**Chapter 3. Configuration and Administration**, reviews components of the landscape and how to make sure your jobs are executed in a successful and timely manner.

**Chapter 4. Application Navigation**, walks new users through a step-by-step process on how to create a simple batch job using each of the development environments in Data Services.

**Chapter 5. Objects**, dives into the primary building blocks that make a job run. Along the way, we provide hints to make jobs run more efficiently and make them easier to maintain over their useful life expectancy and then some!

**Chapter 6. Variables, Parameters, and Substitution Parameters**, explores the process of defining and using variables and parameters. We’ll look at a few examples that use these value placeholders to make your Data Services jobs more flexible and easier to maintain over time.

**Chapter 7. Programming with SAP Data Services Scripting Language and Python**, explores how Data Services makes its toolset extensible through various coding languages and coding objects. Before that capability is tapped, the question, “Why code?” needs to be asked and the answer formulated to ensure that the coding activity is being entered into for the right reasons.

**Chapter 8. Change Data Capture**, is for developers and technical personnel to design and maintain jobs that leverage change data capture (CDC). It starts with what CDC is and the use cases that it supports. This chapter builds a data flow that leverages CDC and ensures a voluminous source table is efficiently replicated to a data warehouse.

**Chapter 9. Social Media Analytics**, explores the text data processing (TDP) capability of Data Services, describes how it can be used to analyze social media data, and suggests more use cases for this important functionality.

**Chapter 10. Design Considerations**, builds on the prior chapters to make sure jobs are architected with performance, transparency, supportability, and long-term total cost in mind.

**Chapter 11. Integration into Data Warehouses**, takes you through data warehousing scenarios and strategies to solve common integration challenges incurred with dimensional and factual data leveraging Data Services. The chapter builds data flows to highlight proper provisioning of data to data warehouses.

**Chapter 12. Industry-Specific Integrations**, takes the reader through integration strategies for the distribution and retail industries leveraging Data Services.

**Chapter 13. SAP Information Steward**, is for the stewardship and developer resources and explores common functionality in Information Steward that enables transparency and trust in the data provisioned by Data Services and how the two technology solutions work together.

**Chapter 14. Where Is SAP Data Services Headed?**, explores the potential future of Data Services.

You can also find the code that’s detailed in the book available for download at [www.sap-press.com/3688](http://www.sap-press.com/3688).

Common themes presented in the book include simplicity in job creation and the use of standards. Through simplicity and standards, we’re able to quickly hand off to support or comprehend from the development team how jobs work. Best practice is to design and build jobs with the perspective that someone will have to support it while half asleep in the middle of the night or from the other side of the world with limited or no access to the development team. The goal for this book is that you’ll be empowered with the information on how to create and improve your organization’s data-provisioning processes through simplicity and standards to ultimately achieve efficiency in cost and scale.
This chapter dives into the primary building blocks that make a job run. Along the way, we provide hints to make jobs run more efficiently and make them easier to maintain over their useful life expectancy and then some!

5 Objects

Now that you know how to create a simple batch integration job using the two integrated development environments (IDEs) that come with SAP Data Services, we dive deeper and explore the objects available to specify processes that enable reliable delivery of trusted and timely information. Before we get started, we want to remind you about the Data Services object hierarchy, illustrated in Figure 5.1. This chapter traverses this object hierarchy and describes the objects in detail.

As you can see from the object hierarchy, the project object is at the top. The project object is a collection of jobs that is used as an organization method to create job groupings.

Note

A job can belong to multiple projects.

We’ll spend the remainder of this chapter discussing the commonly used objects in detail.
When Data Services is considered as an integration solution (and any of its predecessor names such as BODS, BODI, DI, Acta), it’s often only considered as an Extraction, Transformation, Load (ETL) tool where a job is scheduled, runs, and finishes. Although this use case only describes the batch job, in addition to the batch job object type, there also exists a real-time job object type. The job object is really two distinct objects, the batch job object and the real-time job object. We break each of these objects down in detail in the following sections.

### 5.1.1 Batch Job Object

You might be thinking that a “batch” job sounds old, decrepit, and maybe even legacy mainframe-ish. In a way, a batch job is like a legacy mainframe process, as it’s a collection of objects ordered and configured in a way to process information that can be scheduled to execute. That’s where the similarities end. A batch job can also be scaled to perform on multiple servers and monitored in a way that would make any mainframe job jealous. To carry this out, a batch job has properties and has a collection of child objects, including workflows, data flows, scripts, conditionals, while loops, and logs. The properties of the batch job object are discussed in the following subsections.

#### Batch Job Execution Properties

Batch job execution properties enable control over logging, performance, and overriding default values for global substitution parameters and global variables (we’ll go into more detail on this in Chapter 6).

Whether you’re trying to diagnose an issue in an integration process you’re developing, performing QA tests against a job en route to production, or doing root-cause analysis on an issue in production, logging options in batch job execution properties enable insight into what’s occurring while the process, job object, and all other included objects are executing. We discuss the many logging options in the following subsections.

### Monitoring a Sample Rate

This specifies the polling, in seconds, of how often you want logs to capture status about sourcing, targeting, and transforming information. Specifying a small number captures information more often, and specifying a large number consumes fewer system resources and lets more time pass before you’re presented with the errors in the logs.

#### Trace Messages: Printing All Trace Messages versus Specifying One by One

By specifying to PRINT ALL TRACE MESSAGES, you ignore the individually selected trace options. With all traces being captured, the results can be quite verbose, so this option shouldn’t be used when diagnosing an issue, although it does present a simple method of capturing everything. Alternatively, if PRINT ALL TRACE MESSAGES is unchecked, you can specify individually which traces you want to be captured.
Disable Data Validation Statistics Collection
If your job contains data flows with validation transforms, selecting this option will forgo collecting those statistics.

Enable Auditing
If you’ve set audits via Data Services Designer (e.g., a checksum on a field in a table source within a data flow), this will enable you to toggle the capture of that information on and off.

Collect Statistics for Optimization and Use Collected Statistics
By running your batch job with Collect statistics for optimization set, each row being processed by contained data flows has its cache size evaluated. So jobs being executed in production shouldn’t be scheduled to always collect statistics for optimization. After statistics have been collected and the job is executed with the Use Collected Statistics option, you can determine whether to use In Memory or Pageable caches as well as the cache size.

Collect Statistics for Monitoring
This option captures the information pertaining to caching type and size into the log.

Job Performance and Pickup Options
There are also execution options that enable you to control how the job will perform and pick up after a failed run. We discuss these in the following subsections.

Enable Recovery and Recover Last Failed Execution
By selecting Enable Recovery, a job will capture results to determine where a job was last successful and where it failed. Then if a job did fail with the Enable Recovery set, a subsequent run can be performed that will pick up at the beginning of the last failed object (e.g., workflows, data flows, scripts, and a slew of functions).

Job Server or Server Group
A job can be executed from any job server linked with the local repository upon which the job is stored. A listing of these job servers is presented in the drop-down list box. If one or more groups have been specified to enable load balancing, they too are listed as options.

Note
Job servers within a job server group collect load statistics at 60-second intervals to calculate a load balance index. The job server with the lowest load balance index is selected to execute the next job.

Distribution Level
If you’ve selected for the job to be executed on a server group, the job execution can be processed on multiple job servers depending on the distribution level value chosen. The choices include the following:

- **Job Level**
  No distribution among job servers.

- **Data Flow Level**
  Processes are split among job servers to the data flow granularity; that is, a single data flow’s processing isn’t divided among multiple job servers.

- **Sub-Data Flow Level**
  A single data flow’s processing can be divided among multiple job servers.

5.1.2 Real-Time Job Object
Like a batch job, a real-time job object has execution properties that enable differing levels of tracing and some of the execution objects. Unlike a batch job, a real-time job is started and typically stays running for hours or days at a time. Now this doesn’t mean real-time jobs are slow; rather, they stay running as a service and respond to numerous requests in a single run. The rest of this section will show how you create a real-time job, execute a real-time job, and finally make requests.

To create a real-time job from the Data Services Designer IDE, from the Project Area or the Jobs tab of the local repository, right-click, and choose New Real-time Job as shown in Figure 5.2.
The new job is opened with process begin and end items. These two items segment the job into three sections. These sections are the initialization (prior to process begin), real-time processing loop (between process begin and end), and cleanup (after process end) as shown in Figure 5.3.

Within the DF_GetCustomerLocation data flow, you'll notice two sources and one target. Although there can be many objects within a real-time job's data flow, it's required that they at least have one XML Message Source object and one XML Message Target object.

A real-time job can even be executed via Data Services Designer the same way you execute a batch job by pressing the (F8) button (or the menu option DEBUG • EXECUTE) while the job window is active. Even though real-time jobs can be executed this way in development and testing situations, the execution in production implementations is managed by setting the real-time job to execute as a service via the access server within the Data Services Management Console.

**Note**

For a real-time job to be executed successfully via Data Services Designer, the XML Source Message and XML Target Message objects within the data flow must have their XML test file specified, and all sources must exist.
To set up the real-time job to execute as a service, you navigate to the REAL-TIME SERVICES folder, as shown in Figure 5.5, within the access server under which we want it to run.

![Figure 5.5 Real-Time Services](image)

In the REAL-TIME SERVICES window, click the REAL-TIME SERVICES CONFIGURATION tab, and then click the Add button. The window appears filled-in, as shown in Figure 5.6.

![Figure 5.6 Real-Time Service Configuration](image)

After the configuration has been applied, you can start the service by clicking the REAL-TIME SERVICES folder, selecting the newly added service, and then clicking Start (see Figure 5.7). The status changes to Service Started.

![Figure 5.7 Real-Time Services—Start](image)

Now that the service is running, it needs to be exposed. To do this, navigate to the WEB SERVICES CONFIGURATION page, select Add REAL-TIME SERVICE in the combo list box at the bottom, and then click Apply (see Figure 5.8).

![Figure 5.8 Web Services Configuration](image)
The Web Services Configuration – Add Real-Time Services window appears as shown in Figure 5.9. Click Add to expose the web service, and a confirmation status message is shown.

Figure 5.9 Web Services Configuration—Add Real-Time Services

The status of the web service can now be viewed from the Web Services Status tab, where it appears as a published web service (see Figure 5.10). The web service is now ready to accept web service requests.

Figure 5.10 Web Services Status

Note

At this point, you can also view the history log from the Web Services Status page where you can see the status from initialization and processing steps. The clean-up process status isn’t included because the service is still running and hasn’t yet been shut down.

To test the web service externally from Data Services, a third-party product such as SoapUI can be used. After importing the web services’ Web Services Description Language (WSDL) information into the third-party product, a request can be created and executed to generate a response from your real-time job. The sample request and response are shown in Figure 5.11 and Figure 5.12.

Figure 5.11 Sample SOAP Web Service Request

Figure 5.12 Sample SOAP Web Service Response

The rest of this chapter focuses on the remaining Figure 5.1 objects that job objects rely on to enable their functionality and support their processing.

5.2 Workflow

A workflow and a job are very similar. As shown in the object hierarchy, many of the same objects that interact with a job can optionally also interact with a workflow, and both can contain objects and be configured to execute in a specified
order of operation. Following are the primary differences between a job and a workflow:

- Although both have variables, a job has global variables, and a workflow has parameters.
- A job can be executed, and a workflow must ultimately be contained by a job to be executed.
- A workflow can contain other workflows and even recursively call itself, although a job isn’t able to contain other jobs.
- A workflow has the Recover as a unit option, described later in this section, whereas a workflow has the Continuous option, also described later in this section.

You might wonder why you need workflows when you can just specify the order of operations within a job. In addition to some of the functionality mentioned later in this section, the use of workflows in an organization will depend heavily on the organization’s standards and design patterns. Such design patterns might be influenced by the following:

- Organizations that leverage an enterprise scheduling system at times like workflow logic to be built into its streams instead of the jobs in which it calls. In such instances, a rule can emerge that each logical unit of work is a separate job. This translates into each job having one activity (e.g., a data flow to effect the insertion of data from table A to table B). On the other side of the spectrum, organizations that build all their workflow into Data Services jobs might have a single job provision customer from the source of record to all pertinent systems. This latter design pattern might have a separate workflow for each pertinent system so that logic can be compartmentalized to enable easier support and/or re-run ability.
- Organizations may have unique logging or auditing activities that are common across a class of jobs. In such cases, rather than rebuilding that logic over and over again to be included in each job, your best option is to create one workflow in a generic manner to contain that logic so that it can be written once and used by all jobs within that class.

Recommendation

Perform a yearly review of the jobs released into the production landscape. Look for opportunities for simplification where an activity has been written multiple times in multiple jobs. These activities can be encapsulated within a common workflow. Doing so will make future updates to that activity easier to implement and ensure that all consuming jobs of the activity get the update. When creating the workflows to contain the common activity, make sure to denote that commonality with the name of the workflow. This is often done with a CWF_ prefix instead of the standard WF_.

5.2 Areas of a Workflow

A workflow has two tabs, the General tab and the Continuous Options tab. In addition to the standard object attributes such as name and description, the General tab also allows specification of workflow-only properties as show in the following list and in Figure 5.13:

1. **Execution Type**
   - There are three workflow execution types:
     - Regular is a traditional encapsulation of operations and enables sub-workflows.
     - Continuous is a workflow introduced in Data Services version 4.1 (this is discussed in detail in Section 5.2.2).
     - Single is a workflow that specifies that all objects encapsulated within it are to execute in one operating system (OS) process.

2. **Execute only once**
   - In the rare case where a workflow has included a job more than once and only one execution of the workflow is required. You may be wondering when you would ever include the same workflow in the same job. Although rare, this does come in handy when you have a job with parallel processes, and the workflow contains an activity that is dependent on more than one subprocess and varies which subprocess finishes first. In that case, you would include the workflow prior to each subprocess and check the Execute only once checkbox on both instances. During execution, the first time instance to execute will execute the operations, and the second execution will be bypassed.

3. **Recover as a unit**
   - This forces all objects within a workflow to reexecute when the job being executed with the Enable recover option subsequently fails, and then the job is reexecuted with the Recover from last failed execution option. If the job is executed with Enable recover, and the workflow doesn’t have the Recover as a unit option selected, the job restarts in recovery mode from the last failed
object within that workflow (assuming the prior failure occurred within the workflow).

**Bypass**

This was introduced in Data Services 4.2 SP3 to enable certain workflows (and data flows) to be bypassed by passing a value of `YES`. This has a stated purpose to facilitate testing processes where not all activities are required and isn’t intended for production use.

![Figure 5.13 Workflow Object—General Tab](image)

**Figure 5.13** Workflow Object—General Tab

### 5.2.2 Continuous Workflow

Setting the stage, an organization wants a process to wait for some event to occur before it kicks off a set of activities, and then upon finishing the activities, it needs to await the next event occurrence to repeat. This needs to occur from some start time to some end time. To effect a standard highly efficient process, Data Services 4.1 released the continuous workflow execution type setting. After setting the execution type to `Continuous`, the `Continuous Options` tab becomes enabled (see Figure 5.14).

Prior to Data Services 4.1, this functionality could be accomplished by coding your own polling process to check for an event occurrence, perform some activities, and then wait for a defined interval period. Although the same functional requirements can be met without a continuous workflow, the technical performance and resource utilization are greatly improved by using it. As shown in Figure 5.15, the instantiation and cleanup processes only need to occur once (or as often as resources are released), and as shown in Figure 5.16, instantiation and cleanup processes need to occur for each cycle, resulting in more system resource utilization.

![Figure 5.14 Workflow—Continuous](image)

**Figure 5.14** Workflow—Continuous

![Figure 5.15 Continuous Workflow](image)

**Figure 5.15** Continuous Workflow
5.3 Logical Flow Objects

Logical flow objects are used to map business logic in the workflows. These objects are available in the job and workflow workspace. These objects are not available in the data flow workspace. The logical flow objects consist of conditional, while loop, and try and catch block objects.

5.3.1 Conditional

A conditional object can be dragged onto the workspace of a job or workflow object to enable an IF-THEN-ELSE condition. There are three areas of a conditional object:

- Condition
- Workspace area to be executed on the condition resulting in true
- Workspace area to be executed on the condition resulting in false

5.3.2 While Loop

The while loop has two components, a condition and a workspace area. Upon the while condition resulting in true, and while it stays true, the objects within its workspace are repeatedly executed.

5.3.3 Try and Catch Blocks

The try and catch objects, also collectively referred as a try and catch block, refer to two separate objects that are always used in a pair. The try object is dragged onto the beginning of a sequence of objects on a job or workflow, and a catch object is dragged to where you want the execution encapsulation to end. Between these two objects, if an error is raised in any of the sequenced objects, the execution moves directly to the catch object, where a new sequence of steps can be specified.

5.4 Data Flows

Data flows are collections of objects that represent source data, transform objects, and target data. As the name suggests, it also defines the flow the data will take through the collection of objects.

Data flows are reusable and can be attached to workflows to join multiple data flows together, or they can be attached directly to jobs.

There are two types of data flows: the standard data flow and an ABAP data flow. The ABAP data flow is an object that can be called from the standard object. The advantage of the ABAP data flow is that it integrates with SAP applications for better performance. A prerequisite for using ABAP data flows is to have transport files in place on the source SAP application.

The following example demonstrates the use of a standard data flow and an ABAP data flow. We’ll create a standard data flow that has an ABAP data flow that accesses an SAP ECC table, a pass-through Query transform, and a target template table.
The DSF2 Walk Sequencer transform allows mailers to get postal discounts on their mailings. Applying walk sequencing to a mailing sorts the list of addresses into the order the carrier walks their route. The USPS encourages the use of walk sequencing so that delivery operations are more efficient. Prior to using the DSF2 Walk Sequencer transform, the addresses must be CASS certified and delivery point validated (DPV). Both of these conditions can be satisfied using the USA Regulatory Address Cleanse transform.

The DSF2 Walk Sequencer transform enables the following types of postal discounts:

- Sortcode discount
- Walk Sequence discount
- Total Active Deliveries discount
- Residential Saturation discount

### 5.6 Datastores

Datastores connecting to relational databases, as explored in Chapter 3 and Chapter 4, are also able to source and target applications such as SAP Business Warehouse (SAP BW) and a new datastore type introduced in Data Services version 4.2, RESTful web services, which will be explored in this section.

#### 5.6.1 SAP BW Source Datastores

Because different objects are exposed on the SAP BW side for reading and loading data, as shown in Figure 5.109, Data Services uses two different types of datastores:
- SAP BW source datastore for reading data from SAP BW
- SAP BW target datastore for loading data into SAP BW

An SAP BW source datastore is very similar to the SAP application datastore in that you can import the same objects as SAP applications except for hierarchies. SAP BW provides additional objects to browse and import, including InfoAreas, InfoCubes, Operational Datastore (ODS) objects, and Open Hub tables.

There are several steps to setting up a job to read data from SAP BW. We’ll detail them all here.

To set up the RFC server configuration, follow these steps:

1. On the Data Services side, through the Administrator module on the Data Services Management Console, go into SAP Connections • RFC Server Interface.
2. Click the RFC Server Interface Configuration table, and then click the Add button.
3. Enter the information in the fields as shown in Figure 5.110. Click Apply.

The RFC ProgramID is the key identifier between the two systems, and although it can be whatever you want, it needs to be identical between the systems and descriptive. RFC ProgramID is also case sensitive. Username and Password are connection details for SAP BW; it’s recommended that this be a communication user ID and not a dialog user ID. The SAP Application Server name and SAP Gateway Hostname fields correspond with the server name of your SAP BW instance. The Client Number and System Number also correspond with those of your SAP BW instance.

To set up the source system, follow these steps:

1. On the SAP BW side, through the BW Workbench:Modeling, create a new source system by going to Source System • External Source • Create.
2. You'll be prompted for a Logical System Name and Source System Name. (For consistency, we tend to make the Logical System Name the same as the RFC ProgramID we entered for the RFC server configuration. The Source System Name we treat as a short description field.) Click the checkmark button to continue.
3. You'll be presented with the RFC Destination page as shown in Figure 5.111. Click the checkmark button to complete the configuration.

Figure 5.109 SAP Data Services—SAP BW Transactional Layout

Figure 5.110 RFC Configuration in Data Services

Figure 5.111 RFC Configuration in SAP BW
To set up the SAP BW source datastore, follow these steps:

1. In the Data Services Designer, right-click and select New in the Datastore tab of the Local Object Library.

2. Within the Create New Datastore editor, select SAP BW Source as shown in Figure 5.112.

3. Click the Advanced button to open the lower editor window. Under SAP properties, select ABAP execution option (Generate and Execute or Execute Pre-loaded).

4. Enter a Client number and a System number.

5. Select a Data Transfer Method (RFC, Shared Directory, Direct Download, FTP, or Customer Transfer).

6. Depending on the Data Transfer Method, RFC Destination, Working Directory, and Generated ABAP Directory information will also be needed.

7. Click OK to complete the datastore creation.

5.6.2 SAP BW Target Datastore

If you’ll be loading data into SAP BW, there are five main steps, which we discuss in the following subsections:

1. Set up InfoCubes and InfoSources in SAP BW.
2. Designate Data Services as a source system in SAP BW.
3. Create a SAP BW target datastore in Data Services.
4. Import metadata with the Data Services datastore.
5. Construct and execute a job.

Setting Up the InfoCubes and InfoSources in SAP BW with the SAP Data Warehousing Workbench

An InfoSource will be used to hold data that is placed from Data Services. You create and activate an InfoSource using the following steps:
1. In the **Modeling** section of SAP Data Warehousing Workbench, go to the InfoSource window (InfoSources tab).

2. Right-click **InfoSources** at the top of the hierarchy, and select **Create Application Component**. (Application components are tree structures used to organize InfoSources.)

3. Complete the window that appears with appropriate information. So for **APPLICATION COMP**, for example, you might enter "DSAPPCOMP". For **LONG DESCRIPTION**, you might enter "Data Services application component". Press enter.

4. The application component is created and appears in the hierarchy list. Right-click the name of your new application component in the component list, and select **Create InfoSource**.

5. The **Create InfoSource: Select Type** window appears. Select **Transaction data** as the type of InfoSource you’re creating, and press enter.

6. The **Create InfoSource** (transaction data) window appears. Enter the appropriate information, and click enter. The new InfoSource appears in the hierarchy under the application component name.

InfoCubes should be created and activated where the extracted data will ultimately be placed.

**Creating the SAP BW Target Datastore**

To create the target datastore, follow these steps:

1. In the Data Services Designer, right-click and select **New** in the Datastore tab of the Local Object Library.

2. Within the Create New Datastore editor, select SAP BW TARGET as shown on Figure 5.113.

3. Click the **Advanced** button to open the lower editor window.

4. Enter the **Client number** and **System number**.

5. Click **OK** to complete the datastore creation.

**5.6.3 RESTful Web Services**

RESTful Web Services or Representational State Transfer (REST) is a type of web service communication method that allows for the retrieval and manipulation of data through a web server. It relies on standard HTTP operations to select (GET), insert (POST), update (PUT), and delete (DELETE) data. For example, Google provides a web application based on REST to return geographical data for a given input address. The input address is passed to the REST-based web application using HTTP parameters. The following URL calls this web application and specifies Chicago as the value for the HTTP parameter address (address=Chicago). Listing 5.2 shows the resulting geographical information for Chicago returned by the REST service in an XML format.

http://maps.googleapis.com/maps/api/geocode/xml?address=Chicago

```
<?xml version="1.0" encoding="UTF-8"?>
<GeocodeResponse>
  <status>OK</status>
</GeocodeResponse>
```
RESTful web services have enjoyed increased popularity in recent years over other communication methods such as the Simple Object Access Protocol (SOAP). One of the reasons REST has become more popular is because of its simplicity using URL parameters and standard HTTP methods for communicating with the web application to retrieve and manipulate data. In addition, REST supports a wider range of return data formats such as JavaScript Object Notation (JSON), XML, and plain text. REST is also more efficient when communicating with the web application because SOAP requires an XML-formatted request envelope to be sent rather than just HTTP parameters.

### 5.6.4 Using RESTful Applications in Data Services

Beginning with version 4.2, Data Services allows for communicating with a REST-based application through the use of a web service datastore object. The available functions from the REST application can then be imported into the repository and called from within data flows. For example, you may have a need to perform a lookup on key master data from a REST application managing master data. The function call can be done in a Query transform, and the data returned by the REST function call can then be used in your data flow.

To configure a REST web service datastore object, a Web Application Description Language (WADL) file is required. A WADL file is an XML-formatted file that describes the functions and their corresponding parameters available from the REST application. These functions typically use the HTTP methods GET, PUT, POST, and DELETE. Listing 5.3 shows a sample WADL file describing a REST application that has various functions which allow for the retrieval and manipulation of container and item data. Each of the available functions is based on a standard HTTP method.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<application xmlns="http://research.sun.com/wadl/200/10">
  <resources base="http://localhost:9998/storage/">
    <resource path="/containers">
      <method name="GET" id="getContainers">
        <response>
          <representation mediaType="application/xml"/>
        </response>
      </method>
      <method name="DELETE" id="deleteContainer"/>
      <method name="GET" id="getContainer"/>
      <method name="PUT" id="putContainer">
        <response>
          <representation mediaType="application/xml"/>
        </response>
      </method>
      <method name="POST" id="postContainer"/>
      <method name="DELETE" id="deleteContainer"/>
    </resource>
  </resources>
</application>
```

Listing 5.2  Google Maps API Sample XML Output
In addition to the WADL file, the web service datastore objects can be configured for security and encryption as well as configured to use XML- or JSON-formatted data. After the web service datastore has been created, the available functions can be browsed through the Datastore Explorer in the Data Services Designer application. Figure 5.114 shows the view from the Datastore Explorer in Data Services Designer for the same REST application with functions for manipulating container and item data. Notice all seven functions from the WADL file are present along with a nested structure for the functions.

Any of the available functions can then be imported into the repository through the Data Store Explorer. Once imported, they become ready for use in data flows. Figure 5.115 shows the `getItem` function from our WADL file with both a request schema and reply schema defined. Notice the function is based on the HTTP method `GET`. The request schema will include the available input parameters for a function, which in this case, is the parameter `item`. The reply schema will include the XML- or JSON-formatted data in addition to HTTP error codes (`AL_ERROR_NUM`) and error messages (`AL_ERROR_MSG`). These fields can then be integrated and used in downstream data flow processing logic.

5.7 File Formats

As datastore structures provide metadata regarding column attribution to databases and application, file formats provide the same for files. We’ll go over the different file format types and explain how to use them in the following sections.
5.7.1 Flat File Format

To use a flat file as a source or target data source, it needs to first exist in the local repository and visible in the Local Object Library area of the Data Services Designer. The object is a template rather than a specific file. After the object is added to a data flow, then it can be configured to represent a specific file or set of files. We’ll step through the configuration later in this section, but first we’ll create the file format template.

The first step in creating a flat file format template is to right-click Flat Files, and select New as shown in Figure 5.116. The Flat Files objects are located in the Data Services Designer, in the Local Object Library area, in the File Format section.

Creating a Flat File Template

The File Format Editor screen will appear after selecting New as shown in Figure 5.117. This editor will be used to configure the flat file’s structure, data format, and source or target file location. We’ll explain the common configuration steps as we step through examples of creating flat file templates. There are several ways to define the flat file template; one way is to manually create one. To do this, you modify the setting in the left-hand column (Figure 5.117) and define the data structure in the upper-right section.

In the left-hand column, the first option to consider is the type of flat file. Your default choice is Delimited and appropriately is the typical selection. The other choices include Fixed Width, in which the data fields are fixed lengths. These files tend to be easier to read but can be inefficient for storing empty spaces for values smaller than defined widths. SAP TRANSPORTS type is used when you want to create your own SAP application file format. This is the case if the predefined TRANSPORT_FORMAT wasn’t suitable to read from or write to a file in an SAP data flow. The last two types are Unstructured Text and Unstructured Binary. Unstructured Text type is used to consume unstructured files such as text files, HTML, or XML. Unstructured Binary is used to read unstructured binary files such as Microsoft Office documents (Word, Excel, PowerPoint, Outlook emails), generic .eml files, PDF files, and other binary files (Open Document, Corel WordPerfect, etc.). You can then use these sources with Text Data Processing transforms to extract and manipulate data.

The second configuration option is NAME. It can’t be blank, so a generated name is entered by default. Assuming you want a more descriptive name to reference, you’ll need to change the name prior to saving. After the template is created, the NAME can’t be changed.

The third item of the Delimited file is Adaptable Schema. This is a useful option when processing several files, where their formats aren’t consistent. For example,
you're processing 10 files, and some have a comments column at the end, whereas others do not. In this case, you'll receive an error because your schema is expecting a fixed number of elements and finds either more or less. With Adaptable Schema set at Yes, you define the maximum number of columns. For the files with fewer columns, null values are inserted into the missing column placeholders. There is an assumption here that the missing columns are at the end of the row, and the order of the fields isn’t changed.

Creating a Flat File Template with an Existing File

Another method to create a flat file format is to import an existing file. This can be done by using the second section (Data Files) of the left-hand column of the File Format Editor screen. Using the yellow folder icons, select the directory and file to import. You’ll receive a pop-up asking to overwrite the current schema, as shown in Figure 5.118.

The FILE FORMAT EDITOR creates the template based on the data in the specified file and displays the results in the right-hand section as shown in Figure 5.119. Here are a couple of things to remember when using this method:

- The default type of delimiter is Comma. If your results show only one column of combined data, adjust the COLUMN in the DELIMITER section as necessary.
- If the first rows on the files are headers, then you can select to skip row header to change the column names to the headers. If you want the headers written when using the template as a target, then select Yes on the Write Row Header option.
- If you’ll be using the template for multiple files, review the data types and field sizes in the right-hand section. The values are determined from the file you imported; if the field sizes are variable, then ensure the values are set to their maximum. Data types can also be incorrect based on a small set of values; for example, the part IDs in the file may be all numbers, but in another file, they may include alphanumeric values.

Figure 5.119  Imported File Format

Variable File Names

When you move the file format into a data flow as a source, you can edit that object to change a subset of configurations compared to its template. After the name is saved, it can’t be changed on the template or data flow level. The data types can be change only at the template level. In the FILE FORMAT EDITOR in the data flow (Figure 5.120), you can change the file and directory locations, and you
can change the Adaptable Schema setting. You can modify performance settings that weren't present on the template such as Join Rank, Cache, and Parallel process threads. At the bottom of the left-hand section is the Source Information. The Include file name column setting will add a column to the file's data source that contains the file name. This is a useful setting when processing multiple files.

If you intend to process multiple files in a given directory, you can use a wildcard (*). For example, if you want to read all text files in a directory, then the File Name(s) setting would be *.txt. You can also use the wildcard for just a portion of the file name. This is useful if the directory contains different types of data. For example, a directory contains part data and price data, and the files are appended with timestamps. To process only part data from the year 2014, the setting is PART_UPDATE_2014*.

When using the wildcard, Data Services expects a file to exist. An error occurs when the specified file can’t be found. This error is avoided by checking existence prior to reading the flat file. One method of validating file existence is to use the file_exists(filename) function, which given the path to the file will return 0 if no files are located. The one shortcoming of this method is that the file_exists function doesn’t process file paths that contain wildcards, which in many cases is the scenario presented. There is another function, wait_for_file(filename, timeout, interval), which does allow for wildcards in the file name and can be set to accomplish the same results. Figure 5.121 shows the setup to accomplish this. First a script calls the wait_for_file function. Then a conditional transform is used to either process the flat file or catch when no file exists.
5.7.2 Creating a Flat File Template from a Query Transform

The methods to create flat file templates discussed so far can be used for either source or target data sources. Target data sources have one other method to quickly create a flat file template. Within the Query Editor, you can right-click the output schema and select Create File Format as shown in Figure 5.122. This will use the output schema as the data type definition and create a template.

Figure 5.122 Query Transform to Create a Flat File Template

5.7.3 Excel File Format

Excel files are widely used in every organization. From marketing to finance departments, a large number of data sets throughout a company are stored in Excel format. As a result, learning to work efficiently with these in Data Services is critical to the success of many data-provisioning projects.

Excel supports multiple Excel versions and Excel file extensions. The following versions are compatible as of Data Services version 4.2: Microsoft Excel 2003, 2007, and 2010.

Preliminary System Requirements

This section describes the initial step to complete to use an Excel component in Data Services.

\textbf{UNIX Server: Adding an Excel Adapter}

To read the data from an Excel file, you must first define an Excel adapter on the job server in the Data Services Management Console.

\textbf{Warning}

Only one Excel adapter can be configured for each job server.

Follow these steps:

1. Go to the Data Services Management Console.
2. On the left tab under Adapter Instances, click the job server adapter for which you want to add the Excel adapter (see Figure 5.123).
3. On the Adapter Configuration tab of the Adapter Instances page, click Add.

Figure 5.123 Adding a New Excel Adapter

4. On the Adapter Configuration page, enter "BOExcelAdapter" in the Adapter Instance Name field (see Figure 5.124). The entry is case sensitive.
5. All the other entries may be left at their default values. The only exception to that rule is when processing files which are larger than 1MB, in which case, you would need to modify the Additional Java Launcher Options value to -Xms64m -Xmx512 or -Xms128m -Xmx1024m (the default is -Xms64m -Xmx-256m).
6. Start the adapter from the Adapter Instance Status tab (see Figure 5.125).

**Windows Server: Microsoft Office Compatibility**

The 64-bit Data Services Designer and job server are incompatible with Microsoft Office products prior to Office 2010. During installation, if Microsoft Office isn’t installed on the server, the installer will attempt to install Microsoft Access Database Engine 2010. A warning message will appear if an earlier version of Microsoft Office is installed. To remedy this and be able to use Excel workbook sources, upgrade to 2010 64 bit or greater or uninstall Microsoft Office products and install the Microsoft Access Database Engine 2010 redistributable that comes with the Data Services (located in `<LINK_DIR>/ext/microsoft/AccessDatabase-Engine_X64.exe`).

---

**Warning**

Data Services references Excel workbooks as sources only. It’s not possible to define an Excel workbook as a target. For situations where we’ve needed to output to Excel workbooks, we’ve used a User-Defined transform; an example is given in the “User-Defined Transform” section found under Section 5.5.3.

At this point, you can’t import or read a password-protected Excel workbook.

**Importing an Excel File**

To work with Excel workbooks, file layouts need to be defined. To define the layout by importing an existing file, perform the following steps:

1. Go to LOCAL OBJECT LIBRARY area of Data Services Designer.
2. Find the Excel type.
3. Right-click Excel, and select NEW (see Figure 5.126).

An IMPORT EXCEL WORKBOOK dialog window appears as shown in Figure 5.127.
Format Name

Give the component a name in the FORMAT NAME field, and specify the file location (or alternatively, a parameter if one has been defined) and file name as shown in Figure 5.128.

Warning

To import an Excel workbook into Data Services, it must first be available on the Windows file system. You can later change its actual location (e.g., to the UNIX server). The same goes if you want to reimport the workbook definition or view its content.

Figure 5.128 Selecting an Excel File

If the first row of your spreadsheet contains the column names, don’t forget to click the corresponding option as shown in Figure 5.129.

Figure 5.129 First Row as Column Names Option

After the properties have been specified, click the IMPORT SCHEMA button. This will create a file definition based on your range selection. If the column names are present, they’ll be used to define each column, as shown in Figure 5.130. A blank will appear if the content type can’t be automatically determined for a column. When it’s not possible for Data Services to determine the data type, the column will be assigned a default value of varchar(255).

After confirming and updating column attributes if required, click the OK button, and the Excel format object will be saved.

Alternatively, an Excel format can be manually defined by entering the column names, data types, content types, and description. These can be entered in the SCHEMA pane at the top of the NEW FILE CREATION tab. After the file format has been defined, it can be used in any workflow as a source or a target.
Be careful with worksheet names. Blank names or names that contain special characters may not be processed.

Data Access Tab
The Data Access tab, as shown in Figure 5.131, allows you to define how Data Services will access the data if the file isn’t accessible from the job server. Through the tab, you can configure file access through FTP or a custom script, as shown in the following subsections.

FTP
Provide the host name, fully qualified domain name, or IP address of the computer where the data is stored. Also, provide the user name and password for that server. Lastly, provide the file name and location of the Excel file you want to access.

Custom
Provide the name of the executable script that will access the data. Also provide the user name, password, and any arguments to the program.

Note
If both options (FTP/Custom) are left unchecked, Data Services assumes that the file is located on the same machine where the job server is installed.
5.7.4  Hadoop

With the rise of big data, Apache Hadoop has become a widely used framework for storing and processing very large volumes of structured and unstructured data on clusters of commodity hardware. While Hadoop has many components that can be used in various configurations, Data Services has the capability to connect and integrate with the Hadoop framework specifically in the following ways:

- **Hadoop distributed file system (HDFS)**
  A distributed file system that provides high aggregate throughput access to data.

- **MapReduce**
  A programming model for parallel processing of large data sets.

- **Hive**
  An SQL-like interface used for read-only queries written in HiveQL.

- **Pig**
  A scripting language used to simplify the generation of MapReduce programs written in PigLatin.

**Setup and Configuration**

There are two main ways to configure Data Services to integrate with Hadoop. One method is to run your job server on a node within your Hadoop cluster, and the other method is to run the job server and Hadoop on the same machine but outside your Hadoop cluster.

**Prerequisites**

Before attempting to connect Data Services to your Hadoop instance, make sure to validate the following prerequisites and requirements:

- The Data Services job server must be installed on Linux.
- Hadoop must be installed on the same server as the Data Services job server. The job server may or may not be part of the Hadoop cluster.
- The job server must start from an environment that has sourced the Hadoop environment script.
- Text processing components must be installed on Hadoop Distributed File System (HDFS).

**Loading and Reading Data with HDFS**

To connect to HDFS from Data Services, you must first configure an HDFS file format in Data Services Designer. After this file format is created, it can be used to load and read data from an HDFS file in a data flow.

In this example, we’ll connect to HDFS in a **Merge** transform to read files from HDFS.

1. Open Data Services Designer, and create a new data flow. Select the **Formats** tab at the bottom (see Figure 5.132).

![Figure 5.132  Create a New HDFS File Format](image-url)
2. Create a new HDFS file format for the source file (see Figure 5.133).

3. Set the appropriate DATA FILE(s) properties for your Hadoop instance, such as the hostname for the Hadoop cluster and path to the file you want to read in. The HDFS file can now be connected to an object (in this case, a Merge transform) and loaded into a number of different types of targets, such as a database table, flat file, or even another file in HDFS.

In Figure 5.134, we connect HDFS to a Merge transform and then load the data into a database table. The HDFS file can be read and sourced much like any other source object.

5.8 Summary

This chapter has traversed and described the majority of the Data Services Object hierarchy. With these objects at your disposal within Data Services, virtually any data provisioning requirement can be solved. In the next chapter we explore how to use value placeholders to enable values to be set, shared, and passed between objects.
# Contents

Acknowledgments ............................................................................................ 15  
Introduction ..................................................................................................... 17  

PART I  Installation and Configuration  

## 1  System Considerations ................................................................. 23  
1.1  Building to Fit Your Organization ................................................. 23  
1.1.1  Data Services Architecture Scenarios ................................ 24  
1.1.2  Determining Which Environments Your Organization  
       Requires ........................................................................... 29  
1.1.3  Multi-Developer Environment .......................................... 31  
1.1.4  IT and Company Policies .................................................. 33  
1.2  Operating System Considerations ................................................ 36  
1.2.1  Source Operating System Considerations .......................... 37  
1.2.2  Monitoring System Resources .......................................... 37  
1.2.3  CPU ................................................................................. 39  
1.2.4  Memory Considerations .................................................. 40  
1.2.5  Target Operating System Considerations ....................... 41  
1.3  File System Settings ..................................................................... 41  
1.3.1  Locales ............................................................................. 41  
1.3.2  Commands ....................................................................... 42  
1.4  Network ..................................................................................... 45  
1.5  Sizing Appropriately ..................................................................... 45  
1.5.1  Services That Make Up Data Services ............................... 46  
1.5.2  Estimating Usage and Growth .......................................... 52  
1.6  Summary ....................................................................................... 55  

## 2  Installation ................................................................. 57  
2.1  SAP BusinessObjects Business Intelligence Platform and  
    Information Platform Services ...................................................... 57  
2.1.1  Security and Administration Foundation .......................... 57  
2.1.2  Reporting Environment ....................................................... 62  
2.2  Repositories .................................................................................. 64  
2.2.1  Planning for Repositories .................................................... 65  

Acknowledgments ............................................................................................ 15  
Introduction ..................................................................................................... 17  

PART I  Installation and Configuration  

## 1  System Considerations ................................................................. 23  
1.1  Building to Fit Your Organization ................................................. 23  
1.1.1  Data Services Architecture Scenarios ................................ 24  
1.1.2  Determining Which Environments Your Organization  
       Requires ........................................................................... 29  
1.1.3  Multi-Developer Environment .......................................... 31  
1.1.4  IT and Company Policies .................................................. 33  
1.2  Operating System Considerations ................................................ 36  
1.2.1  Source Operating System Considerations .......................... 37  
1.2.2  Monitoring System Resources .......................................... 37  
1.2.3  CPU ................................................................................. 39  
1.2.4  Memory Considerations .................................................. 40  
1.2.5  Target Operating System Considerations ....................... 41  
1.3  File System Settings ..................................................................... 41  
1.3.1  Locales ............................................................................. 41  
1.3.2  Commands ....................................................................... 42  
1.4  Network ..................................................................................... 45  
1.5  Sizing Appropriately ..................................................................... 45  
1.5.1  Services That Make Up Data Services ............................... 46  
1.5.2  Estimating Usage and Growth .......................................... 52  
1.6  Summary ....................................................................................... 55  

## 2  Installation ................................................................. 57  
2.1  SAP BusinessObjects Business Intelligence Platform and  
    Information Platform Services ...................................................... 57  
2.1.1  Security and Administration Foundation .......................... 57  
2.1.2  Reporting Environment ....................................................... 62  
2.2  Repositories .................................................................................. 64  
2.2.1  Planning for Repositories .................................................... 65
## PART III Applied Integrations and Design Considerations

### 9 Social Media Analytics .................................................................383

- 9.1 The Use Case for Social Media Analytics .................................384
  - 9.1.1 It's Not Just Social .........................................................385
  - 9.1.2 The Voice of Your Customer .........................................386
- 9.2 The Process of Structuring Unstructured Data ...........................387
  - 9.2.1 Text Data Processing Overview ....................................387
  - 9.2.2 Entity and Fact Extraction ...........................................388
  - 9.2.3 Grammatical Parsing and Disambiguation ......................393
- 9.3 The Entity Extraction Transform .............................................394
  - 9.3.1 Language Support .......................................................395
  - 9.3.2 Entity Extraction Transform: Input, Output, and Options ..........................396
- 9.4 Approach to Creating a Social Media Analytics Application ........399
  - 9.4.1 A Note on Data Sources ..............................................405
  - 9.4.2 The Data Services Social Media Analysis Project .............407
- 9.5 Summary .................................................................411

### 10 Design Considerations .............................................................413

- 10.1 Performance .....................................................................414
  - 10.1.1 Constraining Results ..................................................414
  - 10.1.2 Pushdown ...............................................................414
  - 10.1.3 Enhancing Performance When Joins Occur on the Job Server ..422
  - 10.1.4 Caching .................................................................423
  - 10.1.5 Degree of Parallelism (DoP) .......................................424
  - 10.1.6 Bulk Loading ...........................................................425
10.2 Simplicity ................................................................. 426
10.2.1 Rerunnable ......................................................... 426
10.2.2 Framework ......................................................... 426
10.3 Summary ................................................................. 427

11 Integration into Data Warehouses ......................... 429
11.1 Kimball Methodology .................................................. 430
11.1.1 Dimensional Data Model Overview ....................... 430
11.1.2 Conformed Dimensions and the Bus Matrix ............. 431
11.1.3 Dimensional Model Design Patterns ....................... 433
11.1.4 Example Orders Star Schema ................................. 437
11.1.5 Processing Slowly Changing Dimensions ................. 438
11.1.6 Loading Fact Tables .............................................. 446
11.2 Hadoop and SAP HANA ............................................. 451
11.3 Summary ................................................................. 454

12 Industry-Specific Integrations ................................. 455
12.1 Retail: Facilitating Customer Loyalty ......................... 455
12.1.1 The Solution ....................................................... 455
12.1.2 Results ............................................................. 464
12.2 Distribution: SAP BW/SAP APO Integration with SAP ECC 465
12.2.1 The Solution ....................................................... 466
12.2.2 Challenges ......................................................... 472
12.2.3 Results ............................................................. 473
12.3 Summary ................................................................. 473

PART IV Special Topics

13 SAP Information Steward ............................................. 477
13.1 Match Review .......................................................... 477
13.1.1 Use Cases for Match Review ................................. 479
13.1.2 Terminology ....................................................... 479
13.1.3 Scenario ........................................................... 481
13.2 Cleansing Package Builder ........................................ 490
13.3 Metadata Management ............................................. 496
13.4 Summary ................................................................. 498

14 Where Is SAP Data Services Headed? ................. 499
14.1 The Key Themes for SAP Data Services ................. 500
14.2 The State of the State .............................................. 501
14.2.1 Simple ............................................................. 502
14.2.2 Big Data ............................................................ 502
14.2.3 Enterprise Readiness .......................................... 502
14.3 Beyond SP03 ......................................................... 504
14.4 Help Shape the Future of Data Services ................. 505
14.5 Summary ................................................................. 508

The Authors ................................................................. 509
Index ............................................................................. 513
## Index

### A

- ABAP data flow, 223
  - create, 225
  - extraction, 229
- ABAP program, dynamically load/execute, 82
- Access server
  - configure, 123, 127
  - parameters, 129
  - SSL, 148
- Accumulating snapshot, 437, 450
- Acta Transformation Language, 493
- Adapter Management, 84
- Adapter SDK, 504
- Adaptive Job Server, 93
- Adaptive Process Server, 93
- Address
  - census data, 81
  - cleansing and standardization, 282
  - data, clean and standardize, 275
  - global validation, 81
  - ISO country codes, 295
  - latitude/longitude, 294
  - list of potential matches, 297
  - street-level validation, 82
- Address Cleanse transform, 74
- Address SHS Directory, 75
- Administrator module, 95
- Adapter Instances submodule, 99
- Central Repository, 100
- Management configuration, 104
- Object Promotion submodule, 101
- real-time job, 97
- Real-Time submodule, 97
- SAP Connections submodule, 98
- Server Group, 100
- Web Services submodule, 98
- Aggregate fact, 437, 450
- Alias, create, 136
- All-world directories, 81
- Apache Hadoop, 324
- Application
  - authorization, 92
  - framework, 46

### Application (Cont.)

- settings, 92

### Architecture

- performance considerations, 24
  - scenario, 24
- system availability/uptime considerations, 24
- web application performance, 27

### Associate transform, 292

### Asynchronous changes, 361

### ATL, export files from Data Services Designer, 32

### Auditing, prevent pushdown, 421

### Authentication, 58

### Auto Documentation

- module, 109
  - run, 109

### B

- BAPI function call, 466
- BAPI function call, read data into Data Services, 470
- Batch job, 208
  - auditing, 210
  - create, 189
  - execution properties, 209
  - logging, 209
  - monitoring sample rate, 209
  - performance, 210
  - statistics, 210
  - trace message, 209
- Batch Job Configuration tab, 97
- Big data, 500
  - unstructured repositories, 451
- BIP, 24, 57, 62, 63
  - CMC, 58
  - licensing, 63
  - patch, 63
  - user mapping, 59
- Blueprints, 404
- BOE Scheduler, 160
- BOE → BIP
- Brand loyalty, 384
Break key, 268
Bulk loading, 425
Bus matrix, 430, 432, 433
Business processes and dimensions, 432
Business transaction, historical context, 360

C

CA Wily Introscope, 131
Caching, 423
in-memory, 374
Case transform, 235, 468
central repository-based promotion, 153
Central Management Console
CMS, 58, 90, 94
Enterprise authentication, 59
deploy and load, 231
datastore configuration, 363
datastore output fields, 365
design considerations, 362
dataflow, 176
data cleansing, 490
cleanse with CDC, 373
consolidation project, 189
de-duplicate, 477
dictionary, 271
extract and load, 231
gather/evaluate from social media site, 384
group, 189
input fields, 272
options, 273
output fields, 272
parsing rules, 272
person standardization options, 273
Data Cleanse transform, 271
cleanup, 271
data formatting options, 275
firm standardization options, 274
input fields, 272
options, 273
output fields, 272
parsing rules, 272
person standardization options, 273
Data Cleansing Advisor, 478
Data flow, 176
add objects, 184
branching paths, 235
bypass, 502
calculate delivery time, 466
cDC tables, 366
configured CDC, 367
create, 184, 198
create with embedded ABAP data flow, 224
definition, 223
delete records, 246
evaluation, 202
fact table loading scenario, 448
flat file error, 169
graphic portrayal, 109
include in batch job, 189
periodic snapshot/aggregate fact table, 450
SCD implementation, 439

D

Data cleansing, 490
calculate with CDC, 373
consolidation project, 189
de-duplicate, 477
dictionary, 271
extract and load, 231
gather/evaluate from social media site, 384
group, 189
input fields, 272
options, 273
output fields, 272
parsing rules, 272
person standardization options, 273
Data Cleanse transform, 271
cleanup, 271
data formatting options, 275
firm standardization options, 274
input fields, 272
options, 273
output fields, 272
parsing rules, 272
person standardization options, 273
Data Cleansing Advisor, 478
Data flow, 176
add objects, 184
branching paths, 235
bypass, 502
calculate delivery time, 466
cDC tables, 366
configure CDC, 367
create, 184, 198
create with embedded ABAP data flow, 224
definition, 223
delete records, 246
evaluation, 202
fact table loading scenario, 448
flat file error, 169
graphic portrayal, 109
include in batch job, 189
periodic snapshot/aggregate fact table, 450
SCD implementation, 439
Global address, single transform, 282
Global digital landscape, 499
Global Postal Directories, 81
Global Suggestion List transform, 297
Global variable, 335
data type, 336

**H**

Hadoop, 451
Hadoop distributed file system (HDFS), 324
Haversine formula, 462
Hierarchical data structure, 265
Hierarchical data structure, 265
IDE, 175, 179
Integration, 30
Integrated development environment, 180
Integration, 30
IPS, 24, 57
cluster installation, 24
CML, 58
licensing, 63
system availability/performance, 24
user mapping, 59

**J**

Job, 176, 208
add global variable, 335
blueprint, 404
common execution failures, 168
common processes, 426
data provisioning, 191
dependency, 160
execution exception, 166
filter, 96
graphic portrayal, 109
processing tips, 426
read data from SAP BW, 300
replication, 191
rerunnable, 426
schedule, 157
specific specific execution, 158
standards, 330
Job Error Log tab, 167
Job execution
log, 166
statistics, 113
Job server, 211
associate local repository, 119
configure, 115
create, 116
default repository, 121
delete, 117
edit, 117
join performance enhancement, 422
remove repository, 120
SSL, 148
Installation
commands, 42
deployment decisions, 62
Linux, 84
settings, 41
Integrated development environment, 180
Integration, 30
IPS, 24, 57
cluster installation, 24
CML, 58
licensing, 63
system availability/performance, 24
user mapping, 59

**K**

Key Generation transform, 249, 441
Kimball approach, 429
principles, 430
Kimball methodology, 429
Klout™ scores, 385
Knowledge transfer, 165

**L**

LACSLink Directory files, 77
Lastline, 276
Latency, 178
Left outer join, 232
Lineage, 64, 496
diagrams, 421
Linux
configure Excel sources, 84
enable Adapter Management, 84
Local repository, 64
Local variable, 337
Log file, retention history, 92
Logical flow object, 275
lookup_ext(), 447, 448

**M**

Mail carrier’s route walk sequence, 79
Management or intelligence tier, 46
Map CDC Operation transform, 259, 366
input data and results, 260
Map Operation transform, 246, 374
inverse ETL job, 246
Map to output, 387
Map, CDC Operation, 467
Map, Operation, 444, 446
transform, 439

Mapping, 200
input fields to content types, 272
multiple candidates, 200
parent and child objects, 264
Master data management, 479
Match results, create through
Data Services, 481
Match Review, 189, 477
approvers/reviewers, 486
best record, 484
configuration, 482
configure job status, 485
data migration, 481
process, 478
results table, 483
task review, 487
terminology, 479
use cases, 479
Match transform, 288, 479
break key, 288
consolidate results from, 292
Group Prioritization option, 288
output fields, 291
scoring method, 289
Memory, 40
page caching, 41
Merge transform, 237
Metadata Management, 64, 496
Mirrored mount point, 25
Monitor log, 97
Monitor, system resources, 37
Multi-developer environment, 65
Multi-developer landscape, 31
Multiline field, 276

**N**

Name Order option, 273
National Change of Address (NCOALink)
Directory files, 80
National Directory, 74
Native Component Supportability (NCS), 131
Natural language processing, 394
Nested data, 263
Nested Relational Data Model (NRDM), 235
NP
**Network traffic**, 45

**New numeric sequence**, 238

---

**Object**, 207

**common**, 426

**create in Data Services Designer**, 186

**hierarchy**, 207

**pairs**, 223

**types**, 175

**Object Promotion**, 155

**execute**, 156

**with CTS+**, 157

**Operating system**, 36

**I/O operations**, 37

**optimize to read from disk**, 37

**Operation code**, UPDATE/DELETE, 372

**Operation Dashboard module**, 113

**Optimized SQL**, 415, 416

**perform insert**, 417

---

**Pageable cache**, 122, 130

**Parameters**, 339

**Parse Discrete Input option**, 274

**Parsing rule**, 271

**Performance metrics and scoring**, 131

**Performance Monitor**, 97

**Periodic snapshot**, 437, 450

**Permissions**, users and groups, 60

**Pipeline forecast**, 465

**Pivot transform**, 267

**Placeholder naming**, 331

**Platform transform**, 231

**Postal directory**, 74, 278

**global**, 41

**IOA**, 74

**Postal discount**, 299

**Postcode directory**, 76

**Postcode reverse directory**, 75

**Predictive analytics**, 385

**Privileges, troubleshoot**, 168

**Product Availability Matrix (PAM)**, 415

**repository planning**, 65

**Production**, 30

**Profiler repository**, 65

**Project**, 176, 180, 191

**create**, 189

**data source**, 192

**lifecycle**, 34

**new**, 192

**Public sector extraction**, 391

**Pushdown**, 414

**Data Services version**, 419

**determine degree of**, 415

**manual**, 421

**multiple datastores**, 415

**prevented by auditing**, 421

**transform instructions**, 419

**Python**, 343, 352

**calculate geographic positions**, 463

**Expression Editor**, 354

**options**, 354

**programming language**, 292

**Python script, analysis project**, 406

---

**Quality assurance**, 30

**Query Editor, mapping derivation**, 187

**Query join, compare data source/target**, 444

**Query transform**, 229, 231

**constraint violation**, 170

**data field modification**, 232

**define joins**, 232

**define mapping**, 186

**filter records output**, 233

**mapping**, 232

**primary key fields**, 235

**sorting criteria**, 234

---

**Range, date sequence**, 249

**Real-time job**, 211, 456

**execute as service**, 214

**execute via Data Services Designer**, 213

---

**Report**, 30

**Reporting environment**, 62

**Repository**, 64

**create**, 67

**create outside of installation program**, 67

**default**, 121

**execution of jobs**, 95

**import ATL**, 493

**manage users and groups**, 100

**number of**, 65

**planning for**, 65

**pre-creation tasks**, 67

**resize**, 119

**sizing**, 66

**update password**, 118

**update with UNIX command line**, 72

**update with Windows command line**, 70

**Repository Manager**, 61

**Windows**, 68

**Representational State Transfer (REST)**, 305

**Residential and business addresses**, 80

**Residential Delivery Indicator**, 60

**Repository planning**, 65

**REST, use applications in Data Services**, 307

**RESTful web service**, 47, 305

**Reverse Soundex Address Directory**, 75

**Reverse-Pivot transform**, 269

**RFC Server configuration**, 300

**RFC Server Interface, add/status**, 98

**Round-robin split**, 425

**Row Generation transform**, 238

**Row, normalized to canonical**, 267

**Row-by-row comparison**, 361

**Rule**, topic subentity, 391

**Rule-based scoring**, 289

**RuleViolation output pipe**, 241

**Runbook**, 165

**Runtime**

**configure**, 130

**edit resource**, 122

---

**Salesforce source tables, CDC functionality**, 377

**Sandbox**, 29

**SAP APO**, 455, 465

**interface, 469

**load calculations**, 467

**read data from**, 470

**SAP BusinessObjects BI platform** → **SAP BusinessObjects Business Intelligence platform**, 24

**SAP BusinessObjects reporting**, 63

**SAP BW**, 465, 469

**load data**, 303

**set up InfoCubes/InfoSources**, 303

**source datastore**, 299, 302

**source system**, 470

**target datastore**, 303, 304

**SAP Change and Transport System (CTS)**, install, 83

**SAP Customer Connection**, 507

**SAP Customer Engagement Intelligence application**, 409

**SAP Data Services roadmap**, 499

**SAP Data Warehousing Workbench**, 303

**SAP ECC**, 455, 465, 466

**data load**, 467

**extraction**, 467

**interfaces**, 471

**SAP HANA**, 451, 505

**SAP HANA, integrate platforms**, 452

**SAP Information Steward** → **Information Steward**

**SAP server functions**, 82

**SCD**, change history, 435

**dimension entities**, 435

**type 1, 438

**type 1 implement in data flow**, 439

**type 2, 441

**type 2 implement in data flow**, 442

**type 3, 443

**type 3 implement in data flow**, 444

**type 4, 444

**type 4 implement in data flow**, 445

**typical scenario, process and load**, 437
Index

Script, 176
obj, 349
Scripting language and Python, 343
Secure Socket Layer (SSL), 142
Semantic disambiguation, 393
Semantics and linguistic context, 388
Sentiment extraction demonstration, 407
Server
  group, 100
list of, 93
Server-based tools, 91
Services configuration, 50
Shared directory, export, 103
Shared object library, 65
SIA
  clustering, 24
  clustering example, 28
  submit, 27
Simple, 500
Simple Object Access Protocol (SOAP), 306
Sizing, 45
Sizing, planning, 54
Slowly changing dimensions (SCD), 434
SMTP
  configuration, 124
  for Windows, 131
Snowflaking, 436
SoapUI, 217
Social media analytics, 383
create application, 404
Voice of the Customer domain, 390
Social media data, filter, 409
Source data, surrogate key, 249
Source object, join multiple, 237
Source system
  audit column, 414
  tune statement, 421
Source table
  prepare/review match results, 478
  run snapshot CDC, 376
Source-based CDC, 360
datastore configuration, 363
SQL query transform, 244
SQL Server database CDC, 369
SQL Server Management Studio, 369
SQL transform, 356
SQL transform, custom code, 356
SQL, execution statements, 415
SSL
  communication via CMS, 143
  configuration for Data Services, 147
  configure, 130
  enable for Data Services Designer Clients, 146
  enable for the Server Intelligent Agent, 143
  enable for web servers, 145
  key and certificate files, 143
  path, 124
  staging, 30
tier, 426
Star schema, 431, 437
dimensional (typical), 433
Start and end dates, 257
Statistic-generation transform object, 113
Street name spelling variations, 75
Structured data, 387
Substitute parameter, 133, 138
Substitution parameter, 331
  configuration, 105
  create, 332
design pattern, 332
  value override, 334
Substitution variable configuration, 171
SuiteLink Directory file, 78
Surrogate key, 435
pipeline, 446
svrcfg, 113
Synchronous changes, 361
System availability, 24
System resources, 37
T
  Table Comparison transform, 254, 361, 372,
    375, 439, 442
  comparison columns, 255
  primary key, 255
  Table, pivot, 267
  Target table, load only updated rows, 367
  Target-based CDC, 360, 372
data flow design consideration, 375
  Targeted mailings, 79
  TDP, 387
dictionary, 390
TDP (Cont.)
  extraction, 388
grammatical parsing, 393
  public sector extraction, 391
  SAP Customer Engagement Intelligence application, 409
  SAP HANA, 410
  SAP Lumira, 404
  semantic disambiguation, 393
Temporary cache file, encrypt, 149
Text Data Processing Extraction Customization Guide, 391
Text data processing → TDP
  Text data, social media analysis, 384
Threshold, Match Review, 482
Tier, 46
Tier to server to services mapping, 49
Time dimension table, 250
Timestamp
  CDC, 361, 376
  Salesforce table, 377
  TNS-less connection, 69
  Trace log, 97
  Trace message, 209
Transaction fact, 437
load, 448
Transform, 176, 231
  compare set of records, 374
  Data Integrator, 249
data quality, 271
  Entity Extraction, 394
  object, 352
  platform, 231
  Transparency, decreased, 421
  Try and catch block, 223
  Twitter data, 408
U
  Unauthorized access, 30
  UNIX, adding an Excel Adapter, 317
Unstructured data, 387
Unstructured text, TDP, 387
Upgrades, 63
US zip codes, 78
US Regulatory Address Cleanse transformation, 275
  add-on mailing products, 279
  address assignment, 275
  address class, 281
  address standardized on output, 279
  CASS certification rules, 280
  Dual Address option, 279
  field class, 281
  map your input fields, 276
  output fields, 281
  Reference Files option group, 278
User
  acceptance, 30
  authenticate, 58
  manage for central repository, 100
  permissions, 92
User acceptance testing (UAT), 31
User-defined transform, 292, 352
map input, 353
USPS
  change of address, 80
discounts, 79
UTF-8 locale, 41
V
  Validation rule, 112
  definition, 241
  full, 241
  failed records, 243
  reporting, 242
  Validation transform, 240
define rules, 241
  Varchar, 202
  Variable, 336
  pass, 339
vs. local variable, 339
  Vertical flattening, 261
  VOC, requests, 390
  Voice of the Customer (VOC) domain, 390
W
  Warranty support, 165
  Web presence, 384
Index

Web server, SSL, 145
Web service, 377
  external testing, 217
  external testing, 217
  history log, 217
Web service datastore object, encryption, 308
Web Services Description Language (WSDL),
  view, 98
Web tier, 46
  clustering, 26
Web-based application, CMC, 91
Weighted average scoring, 290
While loop, 223
Wildcard, 315
Windows cluster settings, 131
Workflow, 176
  conditional object, 222
  continuous, 220
  drill down, 111
  execution types, 219
Workflow (Cont.)
  properties, 219
  versus job, 217

X

XML Map transform, 247, 265
XML Pipeline transform, 263
XML schema, 456, 457
  file format object, 458
  use in data flow, 459
XML_Map, 470, 471
  batch mode, 247
XML_Pipeline transform, 460
XSD, 456
  create, 457

Z

Z4 Change Directory, 78
Bing Chen leads the Advanced Analytics practice at Method360 and has over 18 years of experience in IT consulting, from custom application development to data integration, data warehousing, and business analytics applications on multiple database and BI platforms.

James Hanck is co-founder of Method360 and currently leads the enterprise information management practice.

Patrick Hanck is a lead Enterprise Information Management consultant at Method360. He specializes in provisioning information, system implementation, and process engineering, and he has been recognized through business excellence and process improvement awards.

Scott Hertel is a senior Enterprise Information Management consultant with over 15 years of consulting experience helping companies integrate, manage, and improve the quality of their data.

Allen Lissarrague is a senior BI technologist with over 15 years of experience directly in SAP analytics and information products.

Paul Médaille has worked for SAP for over 15 years as a consultant, trainer, and product manager, and he is currently a director in the Solution Management group covering Enterprise Information Management topics, including Data Services.

We hope you have enjoyed this reading sample. You may recommend or pass it on to others, but only in its entirety, including all pages. This reading sample and all its parts are protected by copyright law. All usage and exploitation rights are reserved by the author and the publisher.