Application Service Providing (ASP) with SAP BW

Version 2.00 - June 2004

Applicable Releases:
SAP NetWeaver ’04, SAP BW 3.5
SAP BW 3.1 Content
SAP BW 3.0B
5.2 Organizational Issues in the BW Administration of an ASP vendor .......................... 49
5.3 Scheduling and Monitoring Loading and Administration Processes .......................... 50
5.4 Availability of Data and Data Quality Management ................................................ 52
5.5 How to Enhance an ASP Scenario with Additional ASP Customers ......................... 56
5.6 Possible Criteria for Charging Operational Costs in a Service Level Agreement (SLA).... 61

6 Summary ......................................................................................................................... 62

7 Appendix ....................................................................................................................... 65
  Appendix 1: Authorizations for Characteristic Values in Input Help .............................. 65
  Appendix 2: Global Transfer Routine for the 0CLIENT InfoObject ............................... 65
  Appendix 3: Customer Exit for a Hierarchy Variable .................................................. 66
  Appendix 4: Copying Process Chains with Generation of Additional InfoPackages ....... 66
  Appendix 5: Code for a Characteristic Value Variable on the 0CLIENT InfoObject ....... 67
  Appendix 6: Code for a Characteristic Value Variable on the 0CLIENT InfoObject when Using SET / GET Parameters ................................................................. 71
  Appendix 7: Reference to SAP Service Marketplace Notes .......................................... 72
1 Introduction

This paper focuses on the tasks of an ASP vendor when implementing SAP BW. It offers best practice solutions for implementing SAP BW in this context.

The general conditions of this analysis are to provide information on the properties of an SAP BW data warehouse in comparison to operative systems, and to examine what an ASP vendor particularly requires from a data warehouse.

After these general observations, support is given in the areas of modeling, authorizations, reporting, performance and administration. However, it is to be noted that this document does not constitute a complete guide to implementing SAP BW, but focuses on particular aspects of an SAP BW project for ASP vendors.

From a technical viewpoint, all observations and implementation tips are based on the following SAP BW releases:

- SAP BW 3.0B SP 21
- SAP BW 3.1C SP 15
- SAP BW 3.5 SP 3

1.1 Data Warehousing with SAP BW

Two issues are particularly significant when designing a data warehouse:
- Creating integrated data models, permitting the optimal information on the basis of an original heterogeneous data basis. The OLAP concept (Online Analytical Processing) is fundamental to this.
- Orienting this data model towards the needs of the end user who requires the information to perform business processes in an optimal manner.

For these reasons, SAP BW provides a range of functions in the area of data warehousing to ensure the extraction, transformation, retention and modeling of relevant data. Furthermore, functions of the BI Platform and BI Suite provide comprehensive and user-friendly options for analysis.
Data models in SAP BW are a customer-specific view of the most important information on the business processes of the customer. This data model can be implemented flexibly with SAP BW by making best possible use of the available Business Content. In contrast, the ability to implement data models flexibly exists to a far lesser extent when implementing an operative system (OLTP = Online Transactional Processing). In this case, the use of standard models for the optimal realization of business processes is often preferred.
As mentioned, a further difference to an operative system lies in how data is integrated. In contrast to an operative system, in which data is kept separate using technical keys (for example, client), a data warehouse aims to integrate master and transaction data. For this reason, the automatic separation of data is avoided in a standard delivery of SAP BW. If you need to separate data on a short or mid-term basis (for example, due to heterogeneous master data), you should consider an enterprise data warehouse strategy or Master Data Management implementation. The consequences these requirements will have on data modeling must be accounted for. You can find more information on this in the white paper “Enterprise Data Warehousing with SAP BW – An Overview“, at [http://service.sap.com/bw](http://service.sap.com/bw).

However, the necessary long-term separation of master and transaction data for an ASP vendor’s end customer can be a data modeling requirement in SAP BW. In this case, ASP vendors will use the flexible implementation of the data model they have designed to ensure the physical separation of the end customer’s data by using separate data models or technical keys. The various technical possibilities within data modeling are discussed in a later chapter.

For more information on the topic “Data Warehousing with SAP BW“, see the “Media Library” directory at [http://service.sap.com/bw](http://service.sap.com/bw).
1.2 Scope of this document

For separating master and transaction data in SAP BW, SAP recommends customers in the SAP environment (most extensive extraction from SAP systems) use the “client” characteristic. This has technical advantages (for example, when using jump targets in an operative system in BW reports) and affords more transparency in the administration of the users and authorizations. This is often already oriented towards clients in scenarios of this kind. In this sense, this document can be seen as a guide to enhancing BW data models flexibly with the client characteristic.

It will be made clear in the course of the document that in certain circumstances a different characteristic can be used as the global criterion for separating master and transaction data in SAP BW, as long as there is no dependency to the SAP source systems (as illustrated above).

However, as this document is primarily concerned with recommendations for application service providing projects with SAP BW, not all aspects of the “client” topic are discussed here. Specifically, this document does not touch on using clients to design test environments in operative systems. In the SAP BW environment, test environments of this kind are not normally built by compounding characteristics (as is recommended for ASP vendors). This is normally done using separate data models and objects in a separate namespace or instance.

A further difference to functionality made available with SAP BW 3.5 in the area of data warehousing must be observed: Multiclient connections with BW-based SAP applications. In a (part) SAP BW-based SAP application (for example, mySAP SCM or SAP MDM), this function makes it possible to address different clients separately during data extraction. This implies only one view of the SAP BW system: SAP BW as the source system. However, this document deals with the more common example that the SAP BW system is the target of the data extraction. Also, from the viewpoint of SAP BW, there is only one technical SAP BW client in a part SAP BW-based application of this kind.
It is also necessary to note that these application-specific tips for SAP BW can be applied, irrespective of whether the BW client whose data model is to be “client-enabled” is being operated on a server or database with other SAP components (in the context of mySAP ERP or SAP NetWeaver, for example).

1.3 Specific Requirements of an ASP Vendor

The following chapter describes the role of application service providers (referred to as ASP vendor in the rest of this document), as well as their specific requirements in the context of an SAP BW implementation.

Generally, an ASP vendor enables other companies or users to access applications via the Internet, without them having to buy or operate these applications themselves. An SLA (Service Level Agreement) outlines the conditions for using this centrally provided software to the end customer. The SLA obliges the application service provider to fulfill certain service conditions. For many companies, working with an application service provider in this way represents an efficient alternative to implementing and operating their own IT solution.

An example of this is small to mid-size utilities companies in Europe who often implement and operate collective IT solutions. This tendency is perhaps more pronounced in this sector, at least in Europe, because of the concept of ‘legal unbundling’: by implementing a liberally organized internal market for energy, the EU plans a separation of vertically integrated utilities companies in terms of their legal form, organization, and power of decision.

In this way, the requirements an application service provider makes on SAP BW reflect the operational and legal environment of their customers, and therefore the requirements
of their customers. Thus, the basic requirements of an ASP vendor when using SAP BW are not different to those of other customers. However, some ASP-specific elements and enhancements have to be considered. These result from the necessity of implementing and operating solutions for different customers in parallel, while also ensuring the level of service stipulated in the contract with the individual customer. Furthermore, there is an additional necessity to implement solutions of very high quality because of this contractually defined relationship between the customer (user in reporting) and operator.

The question of the heterogeneity of an ASP vendor’s customers in terms of their business environment, requirements and, from a technical viewpoint, their data sources, is decisive in how complex the execution is, particularly in the areas of administration and implementation.

In the following section, various requirements are presented in the order of their assumed importance:

### 1.3.1 Aspects of the Authorization Concept

When application service providers implement an SAP BW data warehouse they must ensure that the data for the end customers, who are not perceived as a collective in legal terms, is kept and analysed separately. Therefore, data models that are to be implemented with SAP BW within the scope of the project have to be defined on the basis of organizational (for example, company code) or technical (for example, client) considerations. You will find a more detailed discussion on data modeling with SAP BW in the following chapter.

Furthermore, this separation must also be guaranteed at every stage of reporting, from executing reports (for example, the selection of selection conditions) through to displaying them. Only application service provider employees who undertake overarching tasks in implementing and operating the system (that cannot be broken down by customer), may be exempt from observing this strict separation of data, and then only with express written permission.

### 1.3.2 Operation

The requirements of the application service provider in terms of operating SAP BW largely mirror the basic criteria of the SLA (also see section 5.6). The ASP vendor must do this in the most cost-effective manner possible. The following issues are to be stressed here:

- **Availability of data:** Data has to be provided in SAP BW in the requested reports by a (contractually) agreed point in time. Here, various contractually agreed windows for transferring identical or similar data from one or more source systems may pose a particular consideration for an ASP vendor.
- **Completeness and quality of data:** The end user normally requires a very high quality of data and this takes on even more significance in complex systems such as those managed by ASP vendors, where a large number of reporting-relevant data targets exist. Therefore, automatic checks for errors in the transfer of data that may affect a large number of data targets are essential.
- **Speed of access to reporting in the system:** Finally, the performance of reports executed may not only be directly stipulated in the SLA, but may also indirectly lead to higher costs in terms of hardware and basic operation. Thus, an important
requirement is the ability to execute reports quickly and to a high degree in parallel.

In general, central and efficient monitors are required in all three areas. These monitors facilitate the speedy recognition and resolution of errors and, in this way, enable the ASP vendor to fulfill all contractual obligations and achieve the cost efficiency discussed above. In addition, there should be a proactive process in place to ensure colleagues involved in system operation be made aware of any errors.

1.3.3 Implementation

As well as the (conceptual) data modeling questions that will be clarified later, an application service provider has particular requirements as far as the implementation tool is concerned:

- A development environment that allows the information flow and data models to be developed and transported consistently and in parallel. Lock mechanisms for conjointly-used objects and the administration of namespaces are also to be mentioned in this context.
- Depending on the number and type of customer requests, copy functions may be an important and essential aid.
- The multitude of objects and information flows require a central store for metadata with adequate search functions and where-used lists.
- Finally, it should be possible to simply and centrally implement the authorization concept discussed above both for metadata (separate data models) and at the level of the data (authorization objects).

1.3.4 Reporting

In addition to the authorization concept (which refers to reporting above all), specific requirements for the reporting tool need to be mentioned here, as with implementation. The most important requirement is to provide straightforward access to the requested data. This means:

- An easy-to-use tool for reducing the time and effort incurred through training a large number of users who, in some circumstances, may well be spread across many regions. This also automatically guarantees a reduction in the amount of system support required.
- The simple distribution of reporting tools with no, or as little as possible, installation effort required. Likewise, ‘thin client’ solutions are often desirable in this context.
- Quicker access to centrally stored data. Alternatively, this can also mean a demand for reports to be distributed and made available offline.
2 Modeling and Implementing the Data Models

2.1 Business Blueprint and Data Modeling

An ASP vendor maps the data models of the individual customers from a model-integrative viewpoint in SAP BW. It is important to know which InfoObjects and data targets belong to the general requirements of the ASP customers and which parts of the data model can only be implemented customer specifically.

Changes to the data model of one customer may imply changes for other customers. These changes have to be administered centrally and have to be synchronized for all customers.

This white paper is generally based on the assumption that end customers have the same or at least similar reporting requirements and that these are synchronized so that unique master and transaction data models are implemented. This is generally possible as many ASP vendors already run collective, harmonized, operative OLTP systems for their end users. This scenario can therefore also be adopted when implementing an SAP BW data warehouse.

If heterogeneous requirements of end customers occur, only a decentralized overall data model consisting of separate customer solutions (called “stove pipe solutions”) can be implemented. Some of the best practices quoted in this paper are not applicable in this case as each customer solution is created and implemented separately and therefore corresponds to the logic of many small, separate BW projects.

The following chapter describes the different possible data models and implementations with master data objects, transaction data targets, and in the data flow.
2.2 Data Modeling for Master Data

In the case of customer-specific reporting requirements, separate transaction data targets have to be defined with mostly separate master data objects.

It is possible to use shared objects in the modeling of master data objects, if the reporting requirements are mostly the same. This approach is then preferable because separate master data InfoObjects may involve the following disadvantages (especially where a large number of ASP end customers have to be mapped):

- Shared queries or MultiProviders ( = simplified implementation and maintenance) cannot be used afterwards
- Additional administration of separate master data targets for each end customer. Higher complexity of development and maintenance
- Using Business Content is more difficult

Therefore, different modeling options for shared master data objects are discussed in the following sections of this chapter.

2.2.1 Modeling Shared Master Data Objects

First of all, it is to be mentioned that shared master data objects should be based on Business Content objects wherever possible, in order to reduce implementation costs.

When using shared master data objects, you have to check which InfoObjects have to have their key structure enhanced with an ASP customer-specific component. This will be the case for many objects (for example, business partner, material, cost center and so on), but may be unnecessary for some objects. Not all objects are customer specific, such as value type (plan, actual value, and budget) or time characteristics. These objects can be defined globally without using an additional object in the key. In the case of Business Content time characteristics it is technically even impossible to change the key structure.

You should make the necessary InfoObject key enhancement with one defined single object for all InfoObjects.

- When you predominantly use SAP source systems, the characteristics “client” (InfoObject 0CLIENT) or “source system” (InfoObject 0SOURSYSTEM) are appropriate. The latter is especially recommended to enable a drill-through to reports in an SAP source system.
- When you mainly use non-SAP source systems, you can use any other characteristic as compounding objects (for example “ASP end customer”).

You have two options for the required key enhancement:

2.2.2 Concatenated Key in InfoObjects

The key can be enhanced by a concatenation. This means that the key attribute is prefixed by another key component using a string operation. You have to add these string operations by using the ABAP transformation rules or by using the Transformation Library. Ensure that the technical properties of the key field meet the capacity requirements of the concatenated key.
Example: The concatenated key “MILLER4711” is generated from the material value “4711” of the ASP customer “Miller Inc.”.

This solution is generally used to map a temporary separation of heterogeneous master data (for example, old and new material numbers) within the existing key. You can then avoid compounding to additional objects at database level, thus enabling the long-term integration of this heterogeneous data.

Yet, in most cases, a permanent and secure authorization-based separation of master data characteristics is needed in the ASP environment. Therefore, the concatenated key option is not appropriate. In addition, this solution requires more implementation and administration work than using compounded InfoObjects (see below).

2.2.3 Compounded InfoObjects

Compounded InfoObjects are therefore the most appropriate solution for enhancing the InfoObject key in the ASP environment. This means that the key is enhanced by a new InfoObject. As previously mentioned, a customer-defined object, such as “ASP CUSTOMER” as well as Business Content objects, such as 0CLIENT or 0SOURCESYSTEM, may be used as compound objects.

Best Practice

Shared master data objects are recommended if the data models of the ASP customers are the same or are at least similar. Collective and integrated data targets are only possible where shared master data exists. In addition, it is recommended that ASP vendors separate data in InfoObjects by compounding all InfoObjects to the same InfoObject that is to be defined.

The compounding object is to be authorization relevant because it plays an important role in the implementation of authorizations (see below). In addition, in most cases, you are recommended to maintain the value “unique for every cell” under the InfoObject maintenance in the “selection” field on the Business Explorer tab page. If all SAP BW users can be assigned to one end customer (which may be the case), you can only ever select one single value in the front end from the corresponding compound characteristic (for example a client, company code or a source system).
Please note: For reasons related to the maintenance of authorizations, the global compounding characteristic should be at least 2 characters long (bytes). (For more information, refer to the “Authorizations” chapter).

In order to minimize any additional maintenance for transfer rules, create a transfer routine for the global compound object. This code only needs to be maintained once and is valid for all transfer rules (master and transaction data).

During the data transfer the code that is stored in the individual transfer rule (from transfer structure to communication structure) runs first. Then, the transfer routine for each InfoObject that has a transfer routine and is contained in the communication structure is performed.

This routine is especially useful if the value of the compounding object in transfer rules for master and transaction data cannot be determined from the loaded data, but is determined from the source system. In this case, the value of the global compounding object (for example 0CLIENT, 0COMP_CODE, “ASP customer”) is to be determined in the global transfer routine without creating specific transfer routines for each DataSource.

The following example illustrates how you determine the compounding object “company code” by using the global transfer routine while loading the master data for the characteristic “business partner”.

![Image of a screen displaying a data transfer routine](image-url)
As shown in the diagram, it is often advisable to define a mapping table. This allows a derivation from the source system (here: for the characteristic “company code”). It can be maintained centrally and offers the highest possible level of flexibility.

In the appendix you will find an example implementation of a global transfer routine of this type using a mapping table for the InfoObject 0Client (see appendix 2). This is particularly relevant to the SAP source system environment.

The mapping table could look like this:

**Table ZLOGSYS_CLIENT**

<table>
<thead>
<tr>
<th>LOGSYS (Type LOGSYS)</th>
<th>CLIENT (Type /BIC/OICLIENT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDFCLNT100</td>
<td>100</td>
</tr>
<tr>
<td>SDFCLNT222</td>
<td>222</td>
</tr>
<tr>
<td>SDFCLNT453</td>
<td>453</td>
</tr>
<tr>
<td>RDSCLNT050</td>
<td>100</td>
</tr>
</tbody>
</table>

It should be possible to maintain the entries of this table. It is advisable to save the table RSBASIDOC as a check table for the field LOGSYS. The foreign key RSBASIDOC-SLOGSYS corresponds the LOGSYS field. Using a check table also allows you to use the F4 input help with manual table maintenance. Define the previously mentioned routine after you have activated it in the InfoObject.
If the central compounding object is the characteristic 0SOURSYSTEM and there are separate BW source systems for each end customer, you do not need to define a mapping table of this type because the value of this InfoObject is automatically determined from the source system. But this procedure is only appropriate when it is probable that each ASP customer will never be assigned to more than one source system. In all other cases, a different global compounding characteristic, such as 0CLIENT or 0COMP_CODE, is more useful.

If the value of the global compounding characteristic cannot be determined with a mapping table from the source system name, you should assign the value of the compounding characteristic for each data record in the source data (this means that the value of the compounding object is directly contained in the data structure).

You should also note that compounding is not possible for all characteristics. Standard Business Content time characteristics cannot be changed. Compounding of the objects week, month and year is not even useful because they are valid globally.

However, the fiscal year definition may differ from customer to customer. Here, fiscal year variants are used instead of compounding. It may be necessary to map the same characteristic values for the fiscal year variant (for example “K4”) to different values by using transformation rules.

Ensure that the compounding of all relevant characteristics takes place before the implementation of data flows and transaction data models. It is difficult to add or delete the compounding of an InfoObject after the master data and transaction data have been loaded (since the database has to be converted) and this involves the deletion/rebuild of data. This is mostly the case because of the complex way in which InfoObjects are used in other BW objects.

In view of the following rules for compounding, SAP highly recommends that you predominantly use a global compounding object:

- Transitivity: when A is compounded to B and B to C, then A has to be compounded to C
- Inheritance: when A is compounded to customer and is a navigational attribute of B, then B is also compounded to customer (for example, 0MATL_TYPE as navigational attribute of 0MATERIAL)
- Idempotency: When A is a reference object for B, then both objects have the same compounding (for example 0SOLD_TO und 0CUSTOMER)

**2.2.4 Using Hierarchies**

When you use hierarchies you should take the following aspects into account. They are designed to enable complete security of authorizations for displaying hierarchy nodes in input helps or authorization results (if desired), as well as a simplified handling of hierarchies:

- Separate hierarchies should be used for each ASP customer. With the allocation of technical names, a cross-customer name convention is mandatory. With the connection of separate clients of an SAP source system, the hierarchy for the client/end customer will be transferred separately in any case. SAP does not recommend that you store all ASP customer-specific hierarchies for one object (for example, business partner) in one central hierarchy. This would mean that
you would have to assign reporting authorizations individually on the hierarchy node level using transaction RSSM.

- The authorization to display only specific hierarchies in the Query Designer (input values) or in the display of the report results, is created by using authorization object S_RS_HIER. Assign the corresponding authorizations within an ASP customer-specific role (see 3.1).

- SAP BW allows you to assign authorized objects only to a user. It is recommended that you use hierarchy variables with ASP customer-specific hierarchies in standard queries. These hierarchy variables should not be ready for entry and the hierarchy which is to be displayed in the report should be determined by a customer exit. You will find an implementation example in appendix 3. This is based on the use of the strict naming conventions for hierarchies, mentioned previously. The variable can also be employed for using hierarchy node variables for determining the variables of the underlying hierarchy. You will find the entry “Choose Variable Hierarchy” in the context menu (right mouse-click) for the variable, after having chosen the hierarchy node variable for selection in the Query Designer. You can then select the previously mentioned hierarchy variable in the window.

- It is also important to suppress unassigned nodes, as otherwise customers will find hierarchy nodes from other customers under their unassigned nodes. You do this by setting the hierarchy attribute “suppression of the node ‘unassigned’” in hierarchy maintenance. This flag can be set automatically in the header segment of the transfer rules (InfoObject ONORESTNODE = ‘X’) when loading hierarchies with the transfer method PSA.

- The characteristic for which hierarchies are to be defined has to be flagged as authorization relevant.

The following graphic illustrates the various correlations:
Best Practice

Make sure you have defined an ASP cross-customer naming convention for hierarchies and store hierarchies for each end customer separately. You can use authorization object S_RS_HIER to implement authorization-based access to hierarchies. Avoid displaying unassigned nodes (by maintaining the corresponding hierarchy attribute).

2.3 Data Modeling for Transaction Data

This section deals with the various modeling options for transaction data. It is based on the following assumptions:

- ASP customers obtain their data from a separate source system which is uniquely assigned to them. If this is not the case, this has implications for the modeling of the underlying data flow, but not the modeling of the transaction data model.
- To simplify the process, objects in the data flow of the SAP BW have been omitted. A detailed description of these objects can be found in the “Modeling the Data Flow” chapter.

All modeling options will be evaluated briefly at the end of this chapter. For further details about authorization, reporting, administration and performance, refer to the corresponding chapters in this paper.

2.3.1 Decentralized Data Model

A decentralized data model separates the customer data completely, meaning that there are customer-specific InfoObjects and therefore customer-specific data targets.
This scenario is designed to meet heterogeneous requirements of different ASP customers. The administration work is additive, synergies cannot be used. Do not underestimate the time and effort involved in implementing this. Tasks include copying data targets, transformation rules, aggregation and queries, and maintaining them.

The scenario also involves decentralized master data objects, but should rarely be used, as ideally ASP vendors should seek to offer their customers harmonized out of the box solutions. Such customer-specific individual implementations will be the exception and include higher implementation and administration costs (due to a higher number of objects and lower synergy effects).

The following two modeling options are to be considered where the requirements of all ASP customers are harmonized (as is assumed).

### 2.3.2 Central Data Model

In the central approach, a common data target for all customers is defined. The administration is centralized in this scenario, and this also reduces costs. Compared to the decentralized model, the implementation is also considerably simplified. However, the usually large dimensions of the data targets require specific measures for performance optimization. Furthermore, this scenario offers only limited report creation possibilities for users at the customer. Central master data objects are a prerequisite for this scenario.
2.3.3 Hybrid Data Model

A combination of these two solutions is to provide different data targets for different customers, which can be consolidated later using a MultiProvider. The central query definition is based on a central data model and combined with the “small” data targets (logical partitioning) from the decentralized model. It will be shown later that this model offers optimal flexibility, performance and usability. Again, central master data objects are, however, a prerequisite.

Further details on implementing the central MultiProvider can be found in the section on implementation of MultiProviders.
2.3.4 Overall Evaluation of the Options for Transaction Data Modeling

Best Practice

In many cases, a hybrid data model appears to be the best solution for an ASP vendor considering all the aspects presented in detail in this paper (reporting, authorization, administration and performance). A central solution may be appropriate when the number of end customers is very high and the data volumes for each end customer very small.

In the design phase of the project, your decision should be based on the available sizing information. A decentralized data model is only necessary and recommended, when all the requirements of the end customers are heterogeneous.

Further aspects such as questions about authorization, reporting, administration and performance with the models will be discussed in detail in the corresponding chapters.

2.4 Modeling the Data Flow

The source systems of ASP customers usually also run as an ASP scenario. Rules for implementing the corresponding data flows for extraction will be provided separately as “extraction from SAP source systems” and “extraction from other source systems”.
2.4.1 Extraction from SAP Source Systems

The SAP client concept is appropriate in SAP source systems, such as SAP ERP, for separating data: Each client in the R/3 system (with its logical system name) is connected to SAP BW as individual source system.

If an SAP source system is assigned to several ASP customers, you should take into account that the DataSource as a central BW object for data extraction is cross-client in the SAP source system. Therefore, changes or enhancements to metadata are cross-client as well. This means that it is necessary - as indicated above - to synchronize which information is to be extracted.

The metadata of the DataSource is replicated in SAP BW for each source system. This means that there are multiple (identical) replicates of the DataSources in the BW system.

The following graphic illustrates this.

2.4.2 Extraction from Other Source Systems

As shown, data is transferred to a DataSource for each source system: This means that with SAP source systems the data transfer of a DataSource is client-specific.

When you extract data from non-SAP systems you should always check whether there are separate source systems or DataSources for each end customer. If this is the case, you should expect higher implementation and administration efforts for these data flows.

Ideally only one DataSource, providing data for all end customers, should be used for each business transaction (as is the case with SAP source systems). In contrast to
client-independent SAP source systems, you also have the option of connecting just one source system to an SAP BW.

2.4.3 Data Flow in SAP BW

The DataSources are now assigned in BW to InfoSources, that are independent of the source systems.

Best Practice

It is recommended that you map “identical” DataSources to just one InfoSource as this involves the least amount of outlay in terms of implementation (maintenance and update rules). Furthermore, when you define your update rules for each source system, you should consider the highest possible reusability of the routine to be defined. In general, all transformations which apply across end customers or data targets should be performed in the update rules.

Reusability of transformations can be achieved when using ABAP routines in the transfer rules by making use of function modules or form routines.

If data is updated from the central InfoSource into separate data targets for each end customer, the same applies for the definition of the update rules.

If there are no separate source systems for each end customer, meaning that data for all ASP customers is loaded in one data loading process (called a ‘request’ in BW) from one central source system, the data has to be filtered when it is updated into the data targets (in case of a hybrid data model). It is recommended that you use the start routine in the update rules to filter the data set for each separate data target.

2.5 Implementation Procedure

The following section deals with precise questions about implementation and offers recommendations on how to proceed. Consider the following best practice for all questions raised about the implementation of master data objects:

Best Practice

Any implementation of BW objects should be based on clearly defined requirements and namespaces for each end customer and should be performed for conjointly used objects. It is appropriate to structure the data targets for each end customer into specific InfoAreas. In this way, you can considerably improve the implementation and administration of authorizations.
Based on the recommendation that you should work with compounded InfoObjects in an ASP scenario in the BW environment, you should perform the following steps to implement data models in the development system:

1. Transfer all objects for the relevant Business Content scenarios
2. Determine and maintain the central compounding characteristic (for further details about the maintenance, see above)
3. Decide which master data objects should or can be compounded and compound the central compounding characteristic to these master data objects:
   a. Change all InfoObjects which are affected by the compounding
   b. Transaction RSD1
   c. “Free selection of InfoObjects“ or “All InfoObjects“
   d. Button “Maintain“
   e. Select all required InfoObjects
   f. Add the compounding object in the compounding tab page of the detail view for each InfoObject
   g. Activate all modified objects in a step. To do this you have to return to the previous view, mark all, and choose “Activate“.
4. Add the compounding object to all InfoSources that are used
5. (Optional): You only have to maintain the transfer structures / update rules if the value of the compounding object CANNOT be determined from its global master data routine (See: Details for the maintenance of this characteristic) – for example if the data of all ASP customers is loaded from one BW source system.
6. All data targets that contain one of the activated InfoObjects will be deactivated automatically. The compounding object has to be added in each data target, which then has to be reactivated.
   a. (Optional): You can make the assignment to an end customer-specific InfoArea.
   b. (Optional) When you use a hybrid data model you should assign a constant for the compounding object in InfoCube maintenance. (For further details: See “Implementation of MultiProviders“).
7. Adjust the update rules
   a. Define a 1:1 mapping in the update rules of the InfoSource compounding object to the data target compounding object
   b. Check all routines in the update rules for master data that is being read and that has objects to which compounding has been added. If this is the case, all SELECT queries for the master data table have to be enhanced by the compounding object in the WHERE clause.
8. The compounding object has to be added to all queries that contain an InfoObject that is affected by this. Normally it is added as a filter characteristic.

When you subsequently implement objects you should follow the same procedure. Newly created InfoObjects in particular should be immediately compounded to the central compounding object (if necessary).

If you need to copy objects for each end customer (for example, you need to copy an InfoCube for each end customer in the hybrid scenario), please use the “Create Using Template“ option, which is supported by most object types in SAP BW. The objects that are created with the step-by-step instructions serve as a template. In rare cases you might need to make adjustments (for example, you need to set a constant in an InfoCube for an ASP customer). BAPIs are available for creating and changing BW objects which
you can look at by using the BAPI transaction in your SAP BW system. Further information about this topic can be found in the chapter “Administration”. This describes the process of enhancing an existing ASP scenario to include additional end customers.

2.5.1 Implementing MultiProviders

Note that when using a hybrid data model for transaction data, a constant is to be set for the central compounding object in InfoCube maintenance for each separate InfoCube of an ASP customer. In this way you ensure that authorization filters on the compounding object within central queries (for example, OCLLIENT, 0SOURSYSTEM, 0COMP_CODE, ...) result in the right basis InfoCube to be read by a MultiProvider. In the following example, the company code characteristic is used as a central compounding object:

In InfoCube maintenance, constants are set using menu option “Extras → Structure-Specific Properties of the InfoObject”.
2.5.2 Implementing Source Systems and Source System-Specific Metadata Objects

An ASP vendor generally has a high number of data loading procedures which are to be administered. This is due to the potentially high number of BW source systems. Especially when connecting SAP systems to an SAP BW system, remember that a BW source system has to be created for each client from the SAP system. When you make this connection, which you only need to do once, choose the automatic source system connection, because it saves time.
As mentioned, the metadata (DataSources) created in this SAP source system are cross-client, so they do not have to be created or maintained more than once. Only the replication of DataSources for the (client-specific) source system has to be done for each source system. Note that you have to recreate or modify the transfer rules that are source system dependant after you have replicated a DataSource (when you use an enhancement).

You will find information on procedures for optimally and efficiently adding a new source system of an ASP customer to an SAP BW in the administration chapter under “How to Enhance an ASP Scenario for Additional ASP Customers”.
3 Authorizations

3.1 Requirements and Examination of the Scenarios

As already mentioned, authorization issues in an SAP BW project for application service providers are decisive in the subsequent success of the project. One option for separating data is to create decentralized master and transaction data models. In this approach (described in the ‘modeling’ chapter), data is stored in data models that are already separate. Therefore an authorization concept is only required for the technical names of the metaobjects (InfoCubes, ODS objects, queries), but not for the data itself. Note, however, that where you have a large number of end customers mapped in one SAP BW, this approach can become complex and intensive from an administrative perspective. Therefore it is not the optimal solution.

This chapter aims to explain how optimum security can be provided with authorizations in a central or hybrid data model. The three central aspects, which are shown in the following graphics, are:

- Reporting authorizations: Authorizations for displaying data in reports and (derived from reports, as will be discussed later), and authorizations for displaying data in input help.
  - Example: User is only authorized to display data from company code “3000” in reports or input helps.
- System authorizations: Authorized objects and activities in the front end and backend of SAP BW (including input helps that are made authorization relevant).
  - Example: User can execute queries but is not authorized to create them. An authorization check should be performed on the input help for the “business partner” InfoObject.
- Assignment to menus: Which standard reports are assigned to a user?
  - Example: User has all standard reports from the area controlling assigned.

These three aspects reflect one possible way in which the role concept for end users could be organized.
In what follows we will concentrate on reporting authorizations and system authorizations in the front end.

### 3.2 Authorizations for Displaying Data (Front End)

With authorizations for displaying data, the central object is the InfoObject. This serves as a compounding object for master data characteristics (and can therefore be found in every InfoProvider). This can be the “client” InfoObject or, as in the graphic above, the “company code” InfoObject.

One suggestion of how to implement an authorization of this type follows:

1. Flag the “company code” characteristic as authorization relevant in InfoObject maintenance.
2. Generate an authorization object in transaction RSSM that is to be checked in all InfoProviders and MultiProviders.
3. Only one role can be created per company code (characteristic value, for example, “1000”). Only the appropriate characteristic values are authorized in this role.
4. All reports in SAP BW are now to be filtered with this characteristic in a global filter (Query Designer). For reasons of security and efficiency, basic variables of type “authorization” (that are not ready for input) should be used. These variables allow the relevant data to be made available to the end user in a proactive way.
5. ASP vendor employees are to secure this implementation using central InfoProviders or MultiProviders within the context of creating central reports.

It should be noted that, beyond this central authorization object in reporting, no or very few other authorization objects are defined in reporting. The reason for this is that any increase in complexity (due to multi-dimensional reporting authorizations) causes an increase in administrative efforts and costs. If an end customer has special requirements in terms of reporting authorizations, these should be dealt with separately in an SLA.
Best Practice

Reporting authorization objects are to be defined on the central compounding characteristic (for example, company code). Separate roles are then used to assign authorizations per company code. In queries, authorizations are to be used strictly with authorization variables in a global filter. In this way, you ensure that data content is filtered proactively.

As an alternative to using authorization variables (see point 4 above where the assignment user – client is obtained by assigning the client-specific authorization role), you can also store the assignment user – client (for example, global compounding characteristic) in a mapping table. This is then read using a customer exit variable (instead of an authorization variable). This variable should only allow single values and should be flagged as “not ready for input”. As this variable is used both for filtering in reports and for allocating variable authorizations in authorization objects, its technical name should always be shorter than the length of the characteristic values of the underlying characteristic (for example, 0CLIENT).

The variable is based on a table that contains information on which users are assigned to which clients.

Table ZUSER_CLIENT

<table>
<thead>
<tr>
<th>USER (Type XUBNAME)</th>
<th>CLIENT (Type /BIC/OICLIENT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>JONES</td>
<td>A</td>
</tr>
<tr>
<td>MILLER</td>
<td>B</td>
</tr>
<tr>
<td>BECK</td>
<td>C</td>
</tr>
<tr>
<td>REGIS</td>
<td>A</td>
</tr>
</tbody>
</table>

You can find a description of the corresponding customer exit in appendix 5. If a user is not maintained in this table, no data is displayed in reports. In this context, you must also note that user names have to be defined uniquely in SAP BW (in comparison to operative systems that are separated by client).

Alternatively, you can define a SET/GET parameter ‘CLIENT’ for the /BIC/OICLIENT data element (transaction SE11 – maintaining data elements), and set this in the user master data (“parameter” tab). The code detailed in appendix 6 is relevant here. Furthermore, users must not have the authorization to change their own user master data in this scenario.
3.3 Authorizations for Characteristic Values in Input Help (Front End)

As well as using the authorizations already discussed when displaying data in reports, it can also be appropriate to apply authorizations to input helps in the front end.

The following instances in the Business Explorer require particular mention here:

- Query definition: When displaying all attributes of a characteristic value when selecting available key figures and characteristics for an InfoProvider in the Query Designer.
- Query definition: When selecting filter values in the context menu of a characteristic in the Query Designer
- Query execution: When selecting values for variables after the query has been executed
- When selecting filter values in a navigation block

If you consider that reports are mostly created by ASP vendor employees, the necessity to use authorizations (if desired) is particularly relevant for last two points mentioned.

Because in an ASP implementation all or practically all characteristics are compounded to the central compounding object (as in the example of the company code characteristic), only those characteristic values should be displayed that are valid within the context of the compounding.

A concrete example will now be used to illustrate how you can realize this additional restriction on input help for a concrete characteristic. You do this using the authorization object S_TABU_LIN for maintaining master data in BW.

Note that the authorization objects for maintaining master data and the authorization object for restricting input help are identical in BW 3.x. However, this should not pose problems for reporting users in an ASP scenario, as master data maintenance in BW is very rarely carried out by end users. If this does occur, the user would have restricted authorizations. In our example, as you will see, the user is only authorized to maintain business partners that are compounded to company code 3000.

**Starting Point**

In the example there is a characteristic RHBPART (business partner). This is compounded to characteristic RHCOMPC (company code). The following values exist in the master data table for RHBPART:
Furthermore, the user has a reporting authorization for the authorization-relevant characteristic RHCOMPC for the value ‘3000’. Now, only those business partners that are compounded to company code 3000 are to be displayed in input helps.

**Implementation**

**Generally** you must first take the following steps (this is only necessary once):

1. **Defining a characteristic value variable (here: VCOMPC) for InfoObject RHCOMPC with processing type “customer exit” (mandatory and not ready for input)**

This variable serves to read and relay a user’s reporting authorization in relation to the central compounding object ‘company code’. In this case, the variable uses the customer exit to determine that our user has a reporting authorization for company code 3000. Note: You cannot select a variable with processing type “authorization” here.
2. Implementing the corresponding customer exit in transaction CMOD with enhancement RSR00001.

The ABAP code detailed in appendix 1 is now stored in the corresponding customer include ZXRSU01.

After these two general steps, the **specific steps** follow. These have to be performed **once for each characteristic** (for which input helps are to be restricted). In our case, this is the business partner characteristic (RHBPART).
3. Activating the authorization check for master data maintenance for the characteristic RHBPART.

4. Implementing an authorization with authorization object S_TABU_LIN using variable VCOMPC.

Authorizations for authorization object S_TABU_LIN for characteristic RHBPART should now be allocated in a role that is assigned to the user (ideally the role with system authorizations for the reporting user). The variable implemented earlier (VCOMPC) is now entered with the prefix "$" as the authorized value for the characteristic RHBPART.
In the input help for characteristic RHBPART, the user can see all business partners that are compounded to company code 3000. When the system checks the authorizations for input helps, the variable VCOMPC uses the customer exit and determines company code 3000 (for which the user has reporting authorization). Note that the technical name of the variable must be of the following length:

\[(\text{Length of compounding object} \times 2) - 1 \to 7\], in this case

This is because the FROM and TO fields are available in authorization maintenance for entering variables and because the variable has to be prefixed with the character “$”.

Obviously if you are only implementing an authorization check for input help for a very restricted number of end customers (as in this example: company code), it is also possible to allocate authorizations directly using characteristic values. In our example this would give us “3000/*” instead of “$VCOMPC/*”. In this way you can avoid the third and fourth steps in the implementation process.

An example now follows to illustrate how the values that are displayed from the BEx Query Designer and from the navigation block filter of a BW Web application are filtered:
Note that when implementing this scenario, a known problem may arise. This is documented in SAP Note 667389 and can be resolved with BW 3.0B Support Package 17.

The next slide clarifies the overall process:
Best Practice

You use the authorization object S_TABU_LIN to attain secure input helps for characteristic values in the front end. In order to keep maintenance efforts to a minimum, this should only be implemented for required characteristics (for example, the InfoObject business partner, but not the InfoObject calendar year). The use of variables when allocating authorizations is also recommended in order to keep authorization maintenance efforts to a minimum.

3.4 Authorizations for Hierarchies (Evaluation and Input Helps)

In contrast to (flat) characteristic values, displaying hierarchies in reports and input helps is controlled by a central authorization object. The important point to note when using the assigned authorization object S_RS_HIER have already been discussed in the “Data Modeling for Master Data → “Using Hierarchies” section.

3.5 Report Definition and Execution (Front End)

In addition to authorizations for displaying particular data, system authorizations also have to be considered. System authorizations control both authorized transactions in the system and the editing of InfoObjects. This has to be considered as two separate issues, based on whether it involves the front end (creating queries) or administration. Observations on the functional and organizational side of report definition will be discussed in a subsequent chapter.

SAP generally delivers two templates for authorizations in roles in the front end:
- S_RS_RREPU (reporting user): End user (consumer)
- S_RS_RREDE (reporting developer): Report creators

As the behavior of the user types end user / report creator strongly influences cost in terms of support and training, these aspects have to be considered carefully. This is conditional on the increasing complexity and flexibility of the system when it has to cope with a larger number of user-specific (decentralized) queries.

Best Practice

Authorization for query definition should only be allocated in particular cases and only to the well-trained power users among the ASP vendors’ end customers. A thorough business blueprint and a policy of updating the reports delivered on a cyclical basis on the part of the ASP vendor allows them to exert these restrictions. Authorizations to navigate in reports need not be considered to the same extent, as long as the drilldown and navigation paths contained have been checked previously for the required performance optimizations (for example the use of aggregates). A prerequisite for any type of system authorization is that strict naming conventions are observed (namespace for each end customer) in relation to the meta objects used (InfoCubes, InfoObjects,…).
Depending on the data model, the following conclusions can be drawn in terms of system authorizations in the front end:

<table>
<thead>
<tr>
<th>Data Model</th>
<th>Report Definition</th>
<th>Report Execution</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Decentralized</strong></td>
<td>End customers are only authorized to define reports for their InfoProviders in their InfoAreas. Employees of the ASP vendor can define reports throughout the system.</td>
<td>End customers are only authorized to execute reports for their InfoProviders in their InfoAreas.</td>
</tr>
<tr>
<td><strong>Hybrid</strong></td>
<td>End customers may (if desired) only create reports for their “local” InfoProviders. Only employees of the ASP vendor are authorized to define reports for the central MultiProvider.</td>
<td>End customers execute central MultiProvider reports (the data is filtered – see reporting authorizations). End customers may also execute reports that they have created on assigned local InfoProviders (where these exist).</td>
</tr>
<tr>
<td><strong>Central</strong></td>
<td>As far as possible, reports should be defined by the ASP vendor. Reports created by end customers (in their own namespace) are to be avoided where users are not familiar with the authorization variables used. No data will be displayed without filtering the report-relevant reporting authorizations for the user.</td>
<td>All reports are executed on the central InfoProvider.</td>
</tr>
</tbody>
</table>

This overview illustrates that as far as system authorizations in the front end are concerned, a hybrid data model offers the requisite efficiency and flexibility in implementation (central report definition based on the MultiProvider), while simultaneously guaranteeing the security of authorizations.

### 3.6 Authorizations in Administration (Back End)

If you are concerned with authorizations in the administrative area of SAP BW (only relevant for ASP vendor employees), particular attention has to be paid to the need to offer efficient, comprehensive, and cost-effective support for end customers. How this can be achieved, conceptually and organizationally, is discussed in the “Administration” chapter. The following is true for authorizations in administration:
Best Practice

High-quality back-end support of the solutions offered can only be ensured with extensive authorizations. This concerns both authorizations for data content and for metaobjects. ASP vendor employees involved in support should be subject to a legally binding obligation to maintain confidentiality, within which the extensive possibilities for offering technical support in the system can still be granted. A back-end authorization for employees of the end customer is not provided. One (very restricted) exception may be the maintenance of master data.

3.7 Maintaining Authorizations

You can find more information on maintaining reporting authorizations in SAP BW in the SAP Service Marketplace at http://service.sap.com/bw and on the training course BW365. It is generally advisable to generate authorizations from ODS object data for the automated maintenance of specific reporting authorizations for a large number of end users. You can find information on this procedure in the 'How to… Generate BW Authorization Profiles from a Flat File' document at http://service.sap.com/bw → Services & Implementation → HOW TO… Guides → Guide List SAP BW 3.x.
4 Reporting

The reporting area makes a distinction between report definition and report execution. Report definition concentrates on who creates reports and the system in which they do this.

Issues pertaining to roles and user assignment are fundamentally relevant to report execution (see 3.1).

4.1 Central Report Definition

Central report definition means that the reports are exclusively created by the ASP vendor on behalf of the customer. A requirement for this type of report definition is that reports are usually created only once. They can be used for one or more customers.

This scenario must include a process that an ASP customer can use to inform the ASP of its report requirements. The ASP vendor then implements the report on the basis of the customer’s report specification and makes the report available to the customer that requested it.

In a central report definition scenario, the reports are created in the development system or in the productive system. If reports are created in the development system, they must be transported to the productive system before they can be used.

If you want reports to be created directly in the productive system, you have to set the system change option appropriately. The BW system offers a special feature whereby there are exceptions to BW objects such as queues even if the client settings and system change options do not allow objects to be changed in the productive system (this being the default setting).
As a rule, there is no point in writing query objects to transport requests and transporting them if they are created in the productive system. This means that the transport system is not normally active in the productive system, and so the objects are created as local objects.

Only employees of the ASP vendor have authorization to create reports. Depending on the size of the scenario and of the system, all users might have the same authorizations, or there may be a more refined granularity. The latter would be the case if the ASP vendor wanted certain employees to be responsible for creating HR reports and other employees to be responsible for creating financial reports. Authorizations should be used in such a case to make sure that the two groups of employees can only view and change reports in their area of responsibility.

In principle, a central report definition can be combined with every data model described above. If you want to make it easy for reports to be reused (to be relevant for multiple customers) you should save the reports in a central InfoCube. Because of this, usually only scenarios with central or hybrid transaction data are suitable.

If reports are created in the development system, they must be transported to the productive system before they can be used. Targeted transport strategies that cover at least the following scenarios should be developed for this.

- Continuous development
- Correction of errors
- Emergency corrections

As far as the transport strategy is concerned, a central ASP scenario does not differ from normal BW operation. No special procedures need to be followed.
Best Practice

If you want reports to be created exclusively by ASP vendors, report definition should take place centrally. In order to make reports highly reusable, thereby minimizing maintenance effort, the central report definition should be linked with a central or hybrid transaction data model. In a hybrid transaction data model, report definition takes place in the central MultiCube.

4.2 Decentralized Report Definition

Unlike central report definition, this scenario allows the customers of the ASP vendor to create their own reports. These reports are customer-specific and are not meant to be used by other customers of the ASP vendor.

This means that individual employees of the customer in question need authorizations for creating reports in the system. Employees of one customer should not be able to view or change the reports of another customer.

The decentralized scenario also allows reports to be defined in both the development and the productive system. Creating the reports in the development system greatly influences the transport strategy for the system.

If you want reports to be created directly in the productive system, you have to set the system change option appropriately. The BW system offers a special feature whereby there are exceptions to BW objects such as queues even if the client settings and system change options do not allow objects to be changed in the productive system (this being the default setting).

As a rule, there is no point in writing query objects to transport requests and transporting them if they are created in the productive system. This means that the transport system is not normally active in the productive system, and so the objects are created as local objects.

Because this scenario gives the ASP customer’s employees authorization to create reports, reporting should be handled carefully. The following things should be ensured.

- Customers can only view their own reports
- Customers can only change their own reports
- Naming conflicts between the reports of various customers are avoided
- ...

In other words, the minimum requirement is that separate authorizations for each customer of the provider are given. These authorizations are created at query level, and ensure that the first two requirements are met. The third requirement can be met using naming conventions in connection with authorizations. For example, you can stipulate that customer 1 may only create queries whose technical name begins with the prefix K1. Alternatively, you can use namespaces. This usually involves giving each customer a separate namespace.

Depending on the size of the scenario and of the system, not all employees of a particular customer might have the same authorizations. A more refined granularity may be required. The latter would be the case if the customer of the ASP vendor wanted certain employees to be responsible for creating HR reports and other employees to be responsible for creating financial reports.
In principle, decentralized report definition can be combined with every data model described above. However, since reports are only relevant for one customer and do not need to be reused, a model with decentralized transaction data or a hybrid approach should be used. In the latter case, the reports should not be stored in the central MultiCube; instead they should be stored in the InfoCubes contained therein.

If reports are created in the development system, they must be transported to the productive system before they can be used. Targeted transport strategies that cover at least the following scenarios should be developed for this.

- Continuous development
- Correction of errors
- Emergency corrections

These scenarios are implemented for each customer separately, and this makes the implementation more complex. It is particularly important that customers do not use common query objects such as variables. As far as the transport strategy is concerned, a decentralized ASP scenario does not differ from normal BW operation. No special procedures need to be followed.

**Best Practice**

If you want reports to be created exclusively by ASP customers, report definition should be decentralized. You can isolate processes most successfully if you combine decentralized report definition with a decentralized transaction data model or with a hybrid model. In the latter case, report definition takes place in the separate InfoProviders rather than in the MultiProvider.

### 4.3 Hybrid Report Definition

Hybrid report definition combines the centralized and decentralized approaches. Reports can be created both by the ASP vendor and by customers. The following points should be taken into account:

- The target group should be clearly defined. Only certain users on the customer side should have authorizations to create reports. Every additional user that can define reports gives rise to additional administration tasks.
- Naming conventions. Customer-specific reports should only be created within defined namespaces. This should be ensured using system authorizations (for example, using the authorization object S_RS_COMP).
- InfoProviders versus MultiProviders: Customer-specific reports should only be created in the decentralized, customer-specific InfoProviders. You should also pay attention to the settings for input help authorizations that are mentioned later on (for example, in the InfoCube menu under InfoCube-Specific Properties of InfoObjects).
- Using templates: You can provide default queries as templates for defining customer-specific queries. This helps to ensure quality and usability.
This chapter aims to give practical advice on the administration of a BW system by ASP vendors. Administration is understood as the operation of both the application and the basis system. In particular, this encompasses the following points:

- Evaluating the choice of the data model from the perspective of administration and performance
- Organizational questions in BW administration by an ASP vendor
- Scheduling and monitoring load and administration processes
- Data availability and data quality management
- Procedures and notes for enhancing an ASP scenario to include additional ASP customers
- Possible criteria for settling ongoing costs in a Service Level Agreement (SLA)

Furthermore, certain points are to be considered that are particular to the administration of a BW system by an ASP vendor. Therefore, this paper does not make any claim to cover all aspects of administration and performance, although many of the best practice recommendations are certainly also applicable in other contexts. References to more detailed information on particular topics can be found in each case in connection with particular implementations. You can obtain further detailed information in the training course BW360: “Performance and Administration”.

The areas of activity addressed in this chapter are directly relevant to the operational costs which arise and are settled in the SLA. For this reason, a discussion of the possible costing criteria within an SLA for SAP BW is included at the end of this chapter. Conversely, the conditions defined in an SLA, such as system availability or data validity, are directly relevant to the administrative tasks to be implemented by the ASP vendor. Examples of this include determining maintenance intervals in system administration or the scope of check routines to guarantee the quality of data.

The most important factors influencing the scope of administration within a BW application include the requirements defined in the business blueprint, and how the data model and data flow have been conceived. Based on the assumption that end customers have the same business environments and similar requirements, the following best practice can be deduced:

**Best Practice**

In the business blueprint you should strive to harmonize and standardize the requirements of the end customers. In turn, the data model that is to be conceived and implemented on the basis of this blueprint should then be realized with minimum complexity (for example, with regard to the data flow or authorization aspects).

If this is followed consistently, quick implementation and efficient support of the productive applications can be ensured. This is essential to BW projects in the ASP environment because the necessity of optimizing costs and ensuring high quality performance is stipulated in the SLA.
5.1 Evaluation of overall Data Model from an Administration and Performance Viewpoint

The choice of decentralized, hybrid, or central total data modeling also plays an important role in the administration of SAP BW. The data models discussed above should be considered with respect to the following three most important aspects of BW administration:

- Performance (in particular, runtimes at the front end)
- Timely availability of correct data in data targets
- Administration effort (costs)

The last point makes it clear that a decentralized data model (each customer creates reports separately in separate InfoProviders) needs no further consideration, since the efficient creation and maintenance of reporting is difficult to achieve here (see the section Reporting).

In the hybrid total data model, this significant disadvantage is eliminated by the introduction of central MultiProviders for reporting. The table below compares the remaining options: The hybrid and central total data models.

<table>
<thead>
<tr>
<th></th>
<th>Hybrid data model</th>
<th>Central data model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>+</td>
<td>0</td>
</tr>
<tr>
<td>Administration effort</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>(back end)</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>Administration effort</td>
<td></td>
<td>+</td>
</tr>
<tr>
<td>(front end)</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

Basic Supposition:
End customers have functionally similar requirements or the same business environment
Availability of data per customer

<table>
<thead>
<tr>
<th>Availability of data per customer</th>
<th>+</th>
<th>-</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enhancing a scenario by adding other ASP customers</td>
<td>0</td>
<td>+</td>
</tr>
</tbody>
</table>

This subjective rating is based on the following assumptions:

Performance

- If each customer stores transaction data separately in their own data targets, data is partitioned as far as the models are concerned. This has an extremely positive effect on the overall performance of the reports. After the implementation of aggregates, a central data model can actually attain the same query runtimes (provided that the reporting is structured on InfoCubes). At the runtime of loading and administration processes, another consideration comes into play at the back end: Storing transaction data separately allows the basic parallelization and shortening of administrative processes such as the compression and roll up of aggregates. This is not possible with a central data model.

Administration Effort (Back End)

- Obviously, the smaller number of transaction data targets with a central data model offers an advantage (monitoring and administration). However, an advantage of a hybrid data model is that there will probably be less effort involved in the performance tuning of queries using aggregates.

Administration Effort (Front End)

- If all reports in a hybrid data model are defined in the central MultiProviders, there are no differences here.

Availability of Data per Customer

- On this point, the hybrid data model has significant advantages, because data-target-specific administration (such as correcting data, supporting data loading processes, and short-term performance optimization) can only affect the data availability of one customer.

Enhancing a Scenario by Adding Other ASP Customers

- Extra effort is incurred with a hybrid total data model when setting up additional InfoProviders for new ASP customers. This is normally achieved by means of a copying process, rather than manually. The creation of additional source-system-dependent objects for the new source system is the same in both scenarios. A later section of this document contains information on this.

Of course, all administration measures and the data models to be chosen are dependent on the number of (homogenous) customers to be included in the system and the average data volume per customer. These two parameters determine the necessary performance-optimal total data modeling (separate data targets) and the administration efforts involved. The graphic below shows the relationship between these two criteria and gives a subjective assessment of the best total data model in each case.
ASP Environment and Total Data Model

Number of Customers

Best Practice

To sum up, the hybrid data model is often the best choice for an ASP vendor as far as administration, availability, and performance are concerned. If there are small volumes of data, a very high number of customers, or frequent additions of new ASP customers, central modeling provides a useful alternative.

There are other performance aspects that have significance for ASP vendors but cannot be definitively assigned to one of the described scenarios. The list below gives an overview of some specific challenges that concern the performance tuning of an ASP vendor.

- The database tables of shared master data objects are normally extremely large (especially in the case of customer or material master data, or hierarchies). Because access always takes place via the extended compounding characteristic, this extended key is specified with every access attempt and is also part of the index. Master data accesses (like transaction data accesses) are always qualified by the appropriate ASP customer in ASP scenarios. In order to ensure that performance is good for index accesses, the compounded part always has to be in the first place in the table (or at least, there must be a secondary index).
- Time slot for data transfer: The (possibly high) number of data loading and administration processes must be optimally modeled in process chains. The optimum use of the available time slot can only be achieved through the best usage of parallelization options on process chains and process level.
Authorization checks: Strict authorizations checks are indispensable, especially for the central compounding characteristic. However, you should also remember that the additional implementation of authorization checks on data in reporting or input help can cause an increased runtime in reporting. If such an implementation takes place using customer exists and BADIS (for example, in the case of variables), this must be done in a way that is optimal for performance.

For more detailed information on performance, see http://service.sap.com/bw → Performance.

5.2 Organizational Issues in the BW Administration of an ASP vendor

As a general rule, there are the following three levels of BW support.

- **BW front-end support:**
  - Support for technical questions on reports and creating standard queries. This area is typically handled by the ASP vendor. It is only taken on by the ASP customer under special circumstances.

- **BW back-end support:**
  - Data staging, routine maintenance of data models and data flows, and data-model-specific BW performance. This area is also handled by a BW team belonging to the ASP vendor.

- **BW basis administration:**
  - Administration of Web AS and server and database administration. These tasks are sometimes handled by different people.

The tasks should be shared out as efficiently as possible among the employees involved. In the BW environment, the close interaction of the basis system and the application is particularly important. In general, the data warehouse SAP BW (read access) uses a basis system other than that used by an operative system (write access). This has an effect on authorization issues and required parameterization in the basis area. The flexible implementation of data models means that employees in the BW team (if it is a separate team) often need extra authorizations for basis transactions. The employees of the basis team (if it is a separate team) should take into account the special features of SAP BW when system parameterization takes place.

However, the most important factor when deciding how to share out the tasks is the expertise of each individual involved.

Clear topical areas should be prescribed in a support organization. This can be done by defining dedicated contacts for master data and transaction data models, for example.

All administration tasks should be documented in an operating manual. Note that application-specific issues should also be taken into account in a BW operating manual. Examples of such issues are:

- **Scheduling overview:** Which jobs/process chains run, when do they run, and with what frequency? Are there dependencies or lock situations with other applications that should be taken into account?

- **Who monitors the availability of data in BW data targets, and what tools are used for this?**
• Who monitors the runtime of BW reports? Who is responsible for tracking in each area?

The **maintenance of metadata** should also be considered carefully. If extensions to SAP BW metadata are necessary during the operation of the productive BW solution, note that the BW solution of an SAP provider features a high amount of sharing and a high number of objects.

The following things therefore need to be known:
- Whether the change to be made should or must be specific to one ASP customer.
- Whether the change affects other objects, in particular, objects of other ASP customer (such as in the case of shared master data objects).
- The amount of effort involved in the change.

The required change should be documented in a change request. It is then decided centrally whether or not the change can be carried out. It is recommended that changes are carried out in cycles (in templates) in order to retain optimum stability in the system.

### 5.3 Scheduling and Monitoring Loading and Administration Processes

In general, the number of load processes and administration processes of an ASP vendor depends on the following factors:
- Number of connected source systems: There is a separate load process, scheduled using a separate InfoPackage, for each source system.
- Number of connected DataSources per source system.
- Total data modeling: In a hybrid total data model, administration processes such as the roll up of aggregates are carried out for each InfoCube of one or more ASP customers. Because the data targets are small in comparison to the central data model, there may be less effort required in the area of performance tuning.

It may be that identical InfoPackages need to be created on the same DataSources for different source systems for an ASP vendor. To make sure that this is efficient, you can refer to the BAPI connection for creating InfoPackages. The document *How to …Delete and Create InfoPackages with BAPI.doc*, located at [http://service.sap.com](http://service.sap.com) → “Services & Implementation” → “HOW TO… Guides”, contains further information on this.

Note that when a hybrid total data model is implemented, the ASP-customer-specific data targets should be selected for update in every InfoPackage.

**Best Practice**

Administration and load processes within SAP BW should be implemented and scheduled using BW process chains. When modeling process chains, you have to distinguish between ASP-customer-specific processes and cross-customer processes.

Within the implementation of process chains, the individual processes that are to be carried out should be shared among different process chains. These process chains are
then linked together with meta-chains. The following criteria are available for ASP vendors when sharing processes among process chains.

- Frequency: Daily, weekly, monthly processes
- Whether processes are specific to an ASP customer or are valid for all ASP customers
- Whether processes are for master data or transaction data

When differentiating between ASP-customer-specific and cross-customer processes, it is assumed that each ASP customer has a separate source system. The choice of the transaction data model also plays a part in this question. The table below clarifies this using certain process types as examples.

<table>
<thead>
<tr>
<th>Process type</th>
<th>Hybrid total data model</th>
<th>Central total data model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute InfoPackage</td>
<td>Specific</td>
<td>Specific</td>
</tr>
<tr>
<td>Roll up aggregates</td>
<td>Specific</td>
<td>Cross-customer</td>
</tr>
<tr>
<td>Aggregate InfoCube</td>
<td>Specific</td>
<td>Cross-customer</td>
</tr>
<tr>
<td>Hierarchy and attribute change flow</td>
<td>Cross-customer</td>
<td>Cross-customer</td>
</tr>
</tbody>
</table>

Thus, the weekly load and administration processes of an ASP vendor with a central transaction data model would result in the following structure (simplified here) for the meta-chain mentioned above.

The following things may also be worth considering when implementing the process chains and meta-chains of an ASP vendor.
• Parallelization: The number of parallel process routes in a chain or meta-chain depends on the available system resources.

• Dependency between processes: Whether the links between processes are modeled in red, green, or red/green depends on the relationship between the two processes in question. An important issue here is whether the success of a predecessor process is absolutely necessary for its successor process (for example, a routine involved in loading transaction data reads current master data. This data must already have been loaded successfully).

• Central scheduling of all chains for each meta-chain can only take place if the data of all ASP customers is available in the source system in question at the same time.

• In particular, the hierarchy and attribute change run should be scheduled centrally, because it sets locks in the application.

For more information on modeling process chains, see the presentation *Modeling aspects in process chains* at [http://service.sap.com/bw](http://service.sap.com/bw) → Know How Network.

When it comes to monitoring the BW application, an ASP vendor is particularly likely to implement the central SAP BW monitoring tools. The most important of these are listed below.

• Load and administration processes
  o Monitor and notification functions of the central meta-chains
  o Monitoring of process chains using BW methods in CCMS

• Data availability and data target status
  o Central loading monitor
  o Monitor views of specific BW object types in the Administrator Workbench

• Performance
  o BW statistics content
  o Transaction ST03

For more information on administration and monitoring tools in SAP BW, see the technical operations manual for SAP NetWeaver at [http://service.sap.com/netweaver](http://service.sap.com/netweaver).

### 5.4 Availability of Data and Data Quality Management

Data quality management for an ASP vendor is generally not very different to other requisite aspects of other scenarios in SAP BW. However, two particular aspects have to be considered that necessitate a higher level of caution in the context of ASP vendors:

• An ASP customer receives a guarantee that correct data will be available in SAP BW and this guarantee is legally binding.

• If a central transaction data model is chosen, unavailable or incorrect data can also affect other ASP customers.

If data quality management is to be set up in an ASP vendor’s SAP BW, the following questions have to be asked first:

• Are central data models being used?
• What is the risk of incorrect data?
• How time-consuming and costly is troubleshooting? Can errors be fixed automatically or manually?
• How stable are the connected source systems in terms of data quality?
• What is the cost-value ratio for building additional checks? Do these checks affect load performance and the timely availability of data?

In particular, the following principle is to be observed:

**Best Practice**

Data quality depends on the interaction between the analytical system and the operative (source) system and is not the task of SAP BW alone. Errors are to be corrected at the earliest possible stage of the data flow. Example: If data is updated incorrectly in the operative system this should always be followed by a correction update.

Generally, automatic or customer-specific checks are possible at the following stages in SAP BW 3.x:

**Where: Checks in BW 3.x**

The following checks can be made:
Furthermore, when conceptualizing and implementing data quality checks, the following aspects are to be considered:

- In most cases it is advisable to implement a data warehouse layer in SAP BW for holding granular document data. Updates to further aggregated data targets (for example, InfoCubes) then take place from there. This layer enables data to be validated and subsequently posted in a flexible manner.
- Central documentation on data quality checks and potential sources of error, as well as to what extent these may affect other ASP customers, should be provided.
- Which InfoObjects, master data, and InfoSources are to be checked on the existence of master data (referential integrity) when loading to SAP BW (examples):
  - When the system is to read master data when loading transaction data?
  - When the non-availability of attribute values causes an incorrect picture when reporting using navigation attributes?

Apart from basic technical aids, the following options are available in SAP BW for correcting errors:

- Error handling in SAP BW and the ability to post a subsequent error request which (only) contains the incorrect data records.
- Subsequent posting of a delta requests from the PSA, as long as the cause of the error lies in the data flow of SAP BW.
- Repeat delta requests: With technical faults or errors in the source system the central delta queue in an SAP source system retains the last requested delta records so they can be updated to SAP BW again. The prerequisite for a new request ("repeat delta") is that the last delta load has status "red" in the data load monitor.
“Repair request”: A repair request is possible if data in an overwriting ODS object is only being overwritten with correct values from the source system. You can activate this option for a full upload in an InfoPackage menu.

With a central transaction data model you must consider how to proceed if individual requests cannot be loaded successfully and this affects whether data for other end users is released for reporting.

The following solutions are possible:

- Deleting incorrect requests and rolling up all requests that have not yet been released into aggregates. If no aggregates exist for the InfoCube, any other “green” requests are released directly. Those requests that contain errors can be loaded afterwards from the Persistent Staging Area (PSA) or a pre-loaded ODS object (for example, if using a data warehouse layer).

- As a temporary solution, unreleased requests can be accessed using variable OS_RQALL. For more information on this, see SAP Note 729227: “New Variables for Characteristic 0REQUID”. It is particularly important to note that results in queries that use this variable can only be seen as “preliminary” results. They may differ temporarily from results in other standard queries (that are not using a variable for 0REQUID).

For more information on data quality and availability, see:

- Standard BW documentation
- [http://service.sap.com/bw](http://service.sap.com/bw) → Data Consistency
- [http://service.sap.com/bw](http://service.sap.com/bw) → InfoIndex → Key word „Data Quality“
5.5 How to Enhance an ASP Scenario by Adding Extra ASP Customers

The following section describes how to enhance an existing ASP scenario within a SAP BW and illustrates the implications.

It should generally be assumed that there are approximately as many BW systems as source systems. Example: An ASP vendor who distributes 150 end customers over 3 source systems would try to achieve a similar number of systems to operate the SAP BW system. This estimation is based on the experience that data warehouse systems often have similar or larger data volumes than operational systems. Ultimately, the number of BW systems required is dependant on a detailed hardware sizing with respect to the reporting requirements and data modeling. For example, if analyses should mainly be carried out on an aggregated level, this will positively influence the required hardware landscape.

Furthermore, it should be assumed that runtimes in the data warehouse (especially runtimes of administration processes) will change with an increase in the number of end customers implemented. This is due to an increase in the parallel use of system resources, which might (temporarily) be fixed, as well as in the technical administration activities for monitoring the additional load processes. For further details refer to SAP Note 620361 “Loading Performance Data /Admin. Data Target, Many Requests.”

If you foresee that you will have a large number of end customers to implement when you go live with the first solution, consider the following approaches:

- Implementation of the central transaction data model: In contrast to the hybrid data model, you do not need to implement any additional InfoProviders and integrate them into MultiProviders. This approach has to be checked for data volumes in a central InfoProvider.
- Extensive use of transformations in update rules: When you use central InfoSources which are used across end customer and source systems, you do not necessarily need to adjust the update rules, if additional end users (source systems) are connected to the SAP BW. Transformations in update rules have to be duplicated in this case.
- You should always save fixed references in process variants when you implement process chains. Example: Process variants for “rolling up” aggregates should not reference one InfoPackage (for loading data for an ASP customer), but are always to be based on the InfoCube to be rolled up.
- If this is not the case, the process chains have to be edited as well when the number of ASP customers changes.

The following step-by-step instructions describe how to integrate additional end customers into an existing ASP scenario in the SAP BW with as little effort as possible in as little time as possible. This applies in particular to the implementation of additional source systems or additional source-system-dependent objects in the data flow.
It is assumed that:

- Each ASP customer is assigned uniquely to one source system. The source systems have the same source system type (here: SAP source system).
- The metadata of a DataSource can be used across several systems. In the case of SAP source systems shown in this example, this means that each customer is assigned to the client of an SAP source system. So, the underlying DataSource is cross-client and is already active in the source system if you need to make the enhancement of adding additional clients/end customers.
- During the implementation of process chains, the process chains are divided into customer-specific and cross-system process chains as previously described. For this reason, the data of an ASP customer is loaded in a separate process chain, which (ideally) only contains InfoPackages referencing one source system. Cross-system administration processes for InfoProviders are run in separate, cross-system process chains.
- A central transaction data model has been implemented.

1. Creating a “Template” Transport Request

The transport request which has to be created in the development system should contain the following BW objects for all data flows into a source system:

- Transfer structures
- Transfer rules

The first source system, or one of the source systems that have already been implemented, can serve as a template for the collection. However, it is necessary that the data flows have to be nearly the same or identical for each ASP customer. As long as there are no changes or enhancements in the project or operation of the system, this transport for the implementation of additional end customers can always be re-used (“template transport”).
The BW system provides you with the following support for easily creating this transport request in the transport connection of the Administrator Workbench:

2. Creating the New Source System (End Customer Client in SAP Source System)
   The assumption here, for example, is that this concerns clients in an SAP system. The automatic creation of SAP source systems in the Administrator Workbench is recommended in this case (this is mentioned previously under the area of modeling / implementation). Note that, during this process a user is required in the source system. This user must have been given extensive authorizations. If this is the case, the following steps are performed automatically:
   - An RFC destination and ALE Customizing are created in the source system.
   - A user for the RFC communication is created in the source system (defined in BW using transaction SPRO). For more information about this source system user, see SAP Notes 150315 and 161570.
   - Active DataSource are replicated for this source system (client of the SAP source system).

3. Converting the Logical System Names
   In order to ensure that the objects collected in the transport request are duplicated in the data flow of the new source system when the transport is implemented, table RSLOGSYSMAP has to be maintained. This is in the menu of the Administrator Workbench under Tools → Converting Logical System Names. The original source system corresponds to the source system used in step 1), while the target source system is the system created in step 2).

4. Modifying the Global Transfer Routine for the Global Compound Characteristic
If the global compounding characteristic is not the characteristic 0SOURS SYSTEM, but is supplied with a value indicated to ASP customers using a global transfer routine (see above), the corresponding customer-defined mapping table “Source System → Value of the Compound Characteristic” has to be maintained at this point (see 2.2.3). If desired, steps 3 and 4 can be automated quite easily with ABAP programs.

5. Importing the Transport from 1)

The transport created in step 1 now has to be imported again into the BW development system in which it was created. To facilitate this, appropriate transport paths can be defined in the Transport Management System (transaction STMS). If additional systems in the system landscape (quality assurance or production systems) are also to be provided with the data flow, this transport is still used. After successfully completing this step, complete data flows should exist for the new source system. That is, the transfer structures and transfer rules should exist for the new source system, and be active. For more information on copying formulas and routines in the system correctly, see SAP Note 699317.

6. Creating Process Chains and InfoPackages

So as to facilitate the creation of process chains and InfoPackages for a new ASP-customer source system, a report is delivered as an appendix to this document which is to be created in the customer namespace and offers you the following options:

- Create the process chain for the new ASP customer by copying an existing process chain of another ASP customer. This is to be a process chain that consists exclusively, or to the largest possible extent, of InfoPackages for the source system of the ASP customer. Contained herein are:
  - Definition of a start process for the new process chain
  - Generation of a new (identical InfoPackage) for the newly created source system for the additional ASP customer
  - (Optional): Specification of a prefix so that the newly generated InfoPackage can be distinguished from the old one.

- Note that when you use this report for source system InfoPackages in the process chain only, identical InfoPackages may be generated for the new source system. As this is usually the case, the guidelines specified for allocating process chains should be observed.
The process chain that is now generated can be used in an over-arching metachain. This step completes the process of copying existing objects in the data flow for an enhancement to include new ASP customers.

If you do not have a central transaction data model (as is assumed here) but are using a hybrid transaction data model, additional steps must now be taken:

- Copy the ASP customer-specific InfoProvider
- Copy / recreate the respective update rules
- Enhance central MultiProviders with newly created InfoProviders
- (In some cases): Recreate data mart transfers (export DataSource) if the data from ASP customer-specific InfoProviders are to be updated to further InfoProviders

Finally, any additional systems in the development landscape (for example, the productive system) have to be supplied with the new objects. Do this by following the instructions detailed above. In this case, however, some transports are collected in other ways, and process chains are not copied.

Ultimately, new users have to be created in the BW productive system for the new ASP customers. As SAP BW is operated in a (basis) client, particular care must be taken to ensure that user names are unique.
5.6 Possible Criteria for Charging Operational Costs in a Service Level Agreement (SLA)

In settling operational costs with ASP customers, the question of which (additional) BW-specific criteria are to be considered in an ASP agreement, and how these will be measured in technical terms, must be addressed. The following list offers some benchmarks as suggestions:

- Using standardized data models (Business Content data models in particular) and reports versus customer-specific implementation: Significantly higher costs are to be expected in the second scenario.
- Number of reporting scenarios used: Which standard reports is the customer “buying”?
- Number of users logged on per user type. Here we refer to the categories and assumptions defined in the BW quicksizer (http://service.sap.com/quicksizer):
  - CPU load caused when executing queries is higher than the load caused by data load processes
  - BW user categories according to typical user profiles:
    - Normal user (“Information Consumers”): Normally execute static / pre-defined reports
    - Advanced User (“Executives” or ”Knowledge Workers“): Navigate in reports, use slice and dice, but normally use aggregates
    - Power user (“Analysts“): execute ad-hoc queries and are more likely to have complete access to tables
- Validity of data: the more frequently report results have to be updated, the more frequently transfers of data are triggered (provided that source system data is not accessed directly). These data transfers place a higher demand on administration and use more system resources.
- Volume of data: The higher the volume of data stored for an end customer, the higher the hardware and administration costs. This can be checked for InfoCubes in the BW application using the SAP_INFOCUBES_DESIGNS (content of fact tables) report or using the database transaction DB02
- Number of times queries are accessed: Can be ascertained using technical content (BW Statistics).

The query runtime parameter is only available for “power users” who have more extensive authorizations for query design. Even then, the extent to which customer-specific query design only affects runtime is debateable.
6 Summary

All recommendations made in this document are applicable as of release SAP BW 3.0B. Generally, they are not dependant on notes or support packages. Where they are, they are mentioned separately.

You can weigh the given conceptual and technical questions according to each individual ASP implementation project. The authorization requirements and the number of ASP customers are especially important criteria for determining the optimal implementation approach.

But in general, this document is based on the assumption that the modeling of all requirements in the areas of modeling, reporting and authorizations can be achieved by using BW’s flexible data modeling tools, without a technical (installation) client in SAP BW.

In view of this, this document provides the most important recommendations on how to design and implement “client functionality” or separation of data with regard to authorization security, reporting, performance and administration. Therefore, the following section gives an overview of all previously mentioned best practices.

Business Blueprint and Data Modeling

- This white paper is generally based on the assumption that end customers have the same, or at least similar, reporting requirements and that these can be harmonized so that unique master and transaction data models are implemented. This appears to be possible, as many ASP vendors already run conjoint, harmonized, operative OLTP systems for their end-users.

- Collective master data objects are to be recommended if the data models of the ASP customers are the same or are at least similar. Collective and integrated data targets are only possible where shared master data exists. In addition, it is recommended that ASP vendors separate data in InfoObjects by compounding the InfoObjects into one InfoObject that is to be defined.

- Make sure you have defined an ASP cross-customer naming convention for hierarchies and store hierarchies for each end customer separately. You can use authorization object S_RS_HIER to implement authorization-based access to hierarchies. Avoid displaying unassigned node (by maintaining the corresponding hierarchy attributes).

- In many cases, a hybrid data model appears to be the best solution for an ASP vendor considering all the aspects presented in detail in this paper (reporting, authorization, administration and performance). A central solution be appropriate when the number of end customers is very high and the data volumes for each end customer is very small. In the design phase of a project, your decision should be based on the available sizing information. A decentralized data model is only necessary and recommended, when all the requirements of the end customers are heterogeneous.
- It is recommended that you map DataSources to just one InfoSource as this involves the least amount of outlay in terms of implementation (maintenance and update rules). Furthermore, when you define your update rules for each source system, you should consider the highest possible reusability of the routine to be defined. In general, all transformations which apply across end customers or data targets should be performed in the update rules.

- Any implementation of BW objects should be based upon clearly defined requirements and namespaces for each end customer and should be performed for conjointly used objects. It is appropriate to structure the data targets for each end customer into specific InfoAreas. In this way, you can considerably improve the implementation and administration of authorizations.

**Authorizations**

- Reporting authorization objects are to be defined on the central compounding characteristic (for example, company code). Separate roles are then used to assign authorizations per company code. In queries, authorizations are to be used strictly with authorization variables in a global filter. In this way, you ensure that data content is filtered proactively.

- You use the authorization object S_TABU_LIN to attain secure input helps for characteristic values in the front end. In order to keep maintenance efforts to a minimum, this should only be implemented for required characteristics (for example, the InfoObject business partner, but not the InfoObject calendar year). The use of variables when allocating authorizations is also recommended in order to keep authorization maintenance efforts to a minimum.

- Authorizations for query definition should only be allocated in particular cases and only to the well-trained power users among the ASP vendors’ end customers. A thorough business blueprint and a policy of updating the reports delivered on a cyclical basis on the part of the ASP vendor allows them to exert these restrictions. Authorizations to navigate in reports need not be considered to the same extent, as long as the drilldown and navigation paths contained have been checked previously for the required performance optimizations (for example the use of aggregates). A prerequisite for any type of system authorization is that strict naming conventions are observed (namespace for each end customer) in relation to the meta objects used (InfoCubes, InfoObjects,…).

- High-quality back-end support of the solutions offered can only be ensured with extensive authorizations. This concerns both authorizations for data content and for metaobjects. ASP vendor employees involved in support should be subject to a legally binding obligation to maintain confidentiality, within which the extensive possibilities for offering technical support in the system can still be granted. A back-end authorization for employees of the end customer is not provided. One (very restricted) exception may be the maintenance of master data.

**Reporting**

- If you want reports to be created exclusively by ASP vendors, report definition should take place centrally. In order to make reports highly reusable, thereby minimizing maintenance effort, the central report definition should be linked with a central or hybrid transaction data model. In a hybrid transaction data model, report definition takes place in the central MultiCube.
• If you want reports to be created exclusively by ASP customers, report definition should be decentralized. You can isolate processes most successfully if you combine decentralized report definition with a decentralized transaction data model or with a hybrid model. In the latter case, report definition takes place in the separate InfoProviders rather than in the MultiProvider.

Administration and Performance

• In the business blueprint you should strive to harmonize and standardize the requirements of the end customers. In turn, the data model that is to be conceived and implemented on the basis of this blueprint should then be realized with minimum complexity (for example, with regard to the data flow or authorization aspects).

• To sum up, the hybrid total data model is often the best choice for an ASP vendor as far as administration, availability, and performance are concerned. If there are small volumes of data, a very high number of customers, or frequent additions of new ASP customers, central modeling provides a useful alternative.

• Administration and load processes within SAP BW should be implemented and scheduled using BW process chains. When modeling process chains, you have to distinguish between ASP-customer-specific processes and cross-customer processes.

• Data quality depends on the interaction between the analytical system and the operative (source) system and is not the task of SAP BW alone. Errors are to be corrected at the earliest possible stage of the data flow. Example: If data is updated incorrectly in the operative system this should always be followed by a correction update.
Appendix 1: Authorizations for Characteristic Values in Input Help

Implementing the customer exit in transaction CMOD with enhancement RSR00001. The following ABAP coding is stored in the associated customer include ZXRSU01:

```abap
* &---------------------------------------------------------------------*
* &  Include           ZXRSRU01                                         *
* &---------------------------------------------------------------------*

data: l_s_range        type rsr_s_rangesid,
     l_s_rangesid     type rsr_t_rangesid,
     l_rangesid       like rrrangesid.

case i_vnam.
  when 'VCOMPC'.
    IF i_step = 0 or i_step = 1.
      CALL_FUNCTION 'RSSB_AUTHORIZATIONS_OF_USER'
        EXPORTING
          I_IOBJNM          = 'RHCOMPC'
          I_INFOPROV        = i_s_rkb1d-infocube
          I_UNAME           = SY-UNAME
        IMPORTING
          E_T_RANGESID      = l_s_rangesid
      EXCEPTIONS
        NOT_AUTHORIZED    = 1
        INTERNAL_ERROR    = 2
        USER_DOESNT_EXIST = 3
        X_MESSAGE         = 4
        OTHERS            = 5.
      IF SY-SUBRC = 0.
        loop at l_s_rangesid into l_rangesid.
          IF l_rangesid-low(1) <> '$'.
            move-corresponding l_rangesid to l_s_range.
            append l_s_range to e_t_range.
          ELSE.
            EXIT.
          ENDIF.
        endloop.
      ELSE.
        EXIT.
      ENDIF.
    ENDIF.
  when others.
endcase.
```

Appendix 2: Global Transfer Routine for the 0CLIENT InfoObject

This routine is to show, for example, how a customer-defined table (ZLOGSYS_CLIENT with fields LOGSYS and CLIENT) can be used in the global transfer routine of the 0CLIENT InfoObject to determine the corresponding value of the 0CLIENT characteristic
from the source system. A similar logic is also to be employed as a global compounding characteristic when using other InfoObjects (for example, 0COMP_CODE).

PROGRAM CONVERSION_ROUTINE.
* Type pools used by conversion program
TYPE-POOLS: RSD, RSARC, RSARR.

TYPES: DE_0CLIENT(000003) TYPE N.

* Conversion rule for InfoObject 0CLIENT
* Data type = NUMC
* ABAP type = N
* ABAP length = 000003
FORM CONVERT_0CLIENT
USING
SOURCE_SYSTEM TYPE RSA_LOGSYS
IOBJ_NAME TYPE RSIOBJNM
CHANGING RESULT TYPE DE_0CLIENT " InfoObject value
RETURNCODE LIKE SY-SUBRC.

DATA: L_ZLOGSYS_CLIENT TYPE ZLOGSYS_CLIENT.
SELECT SINGLE * FROM ZLOGSYS_CLIENT INTO L_ZLOGSYS_CLIENT
WHERE LOGSYS = SOURCE_SYSTEM.
IF SY-SUBRC = 0 AND
NOT ( L_ZLOGSYS_CLIENT-CLIENT IS INITIAL ).
* client found
RESULT = L_ZLOGSYS_CLIENT-CLIENT.
RETURNCODE = 0.
ELSE.
* no client found in table ZLOGSYS_CLIENT --> error
RETURNCODE = 4.
ENDIF.
ENDFORM.

Appendix 3: Customer Exit for a Hierarchy Variable

The following code can serve as a template for creating a customer exit for a hierarchy variable. In the example given here, the ASP customer is mapped using the 0CLIENT characteristic. Upon executing the query, a customer-defined mapping table is used to select the value from 0CLIENT that is assigned to the system user. This value is then used to select the correct ASP customer-specific hierarchy.

CASE i_vnam.             " <== This CASE statement is for ALL variables
WHEN ' <TECHNICAL NAME OF HIERARY NAME VARIABLE>'.
  * get client from user name
    DATA: l_client TYPE /bic/oiclient.
    REFRESH e_t_range.
    SELECT SINGLE client FROM  zuser_client INTO (l_client)
    WHERE uname = sy-uname.
    IF sy-subrc = 0.
    * user found
      l_s_range-sign     = 'I'.
      l_s_range-opt      = 'EQ'.
      CONCATENATE 'HIER_' l_client INTO l_s_range-low
      l_s_range-high     = ''.
      APPEND l_s_range TO e_t_range.
    ENDIF.
  ELSE.
  * no client found in table ZLOGSYS_CLIENT --> error
    RETURNCODE = 4.
  ENDIF.
ENDCASE.
Appendix 4: Copying Process Chains with Generation of Additional InfoPackages

* &---------------------------------------------------------------------* 
* & Report   ZCOPYCHAINIP                                                * 
* &                                                                     * 
* &---------------------------------------------------------------------* 
* &                                                                     * 
* &                                                                     * 
* &---------------------------------------------------------------------* 

PREREQUISITES
1) ONLY ONE SOURCE SYSTEM PER PROCESS CHAIN GETS CONVERTED
2) IDENTICAL DATASOURCE EXISTS FOR TARGET SOURCE SYSTEM
3) BOTH SYSTEMS HAVE SAME SOURCE SYSTEM TYPE
4) ONLY FIX REFERENCES IN VARIANTS OF OTHER PROCESSES

REPORT  ZCOPYCHAINIP.

PARAMETERS: copypc TYPE rspc_chain,
sysfrom TYPE RSSLOGSYS,
systo TYPE RSSLOGSYS,
prefix(10) type c.

DATA: I_T_INFOCUBES          type BAPI6109IC occurs 0,
I_T_THIRD_PARTY_PARAMS type BAPI6109TCP occurs 0,
I_S_FILE_PARAMS        type BAPI6109FILE,
I_S_HIE_PARAMS         type BAPI6109HIE,
I_S_DETAIL_LS          type BAPI6109,
I_S_SCHEDULING_PARAMS  type BAPI6109BTCH,
I_S_DESTINATIONS       type BAPI6109DEST,
I_S_SELECTIONS         type BAPI6109,
I_T_RETURN             like BAPIRET2 occurs 0,
I_S_RETURN             like BAPIRET2,
I_T_SELECTIONS         type BAPI6109SEL occurs 0.

DATA: l_r_chain    TYPE REF TO cl_rspc_chain,
l_t_chain    TYPE TABLE OF rspcchain,
l_h_chain    TYPE rspcchain,
l_s_chain    TYPE rsppchain,
l_tabix      TYPE sytabix,
l_chain      TYPE rspc_chain,
l_s_trigger  TYPE rspc_s_chain,
l_infopackage LIKE BAPI6109-INFOPACKAGE,
t_infopackage LIKE BAPI6109-INFOPACKAGE,
l_text(70) type c.

DATA: BEGIN        OF   l_s_ipaks,
        ur_logdpid   TYPE rslogdpid,
        ziel_logdpid TYPE rsllogdpid,
END          OF   l_s_ipaks,
DATA: OF   l_t_ipaks OF l_s_ipaks OCCURS 0.

CALL FUNCTION 'BAPI_SRCSYSTEM_GETDETAIL' EXPORTING
   SOURCESYSTEM        = sysfrom
   IMPORTING
   TEXTLONG            =
   DWSYSTEM            =
   BASICIDOCTYPE       =
   PORTNAME            =
   SYSTEMTYPE          =
   RETURN              = l_s_RETURN
   METHOD              =
* PROGRAM =
* GWHOST =
* GWREGION =
* HOST =

IF I_S_RETURN IS NOT INITIAL.
  MESSAGE E051(rsar) WITH 'Source system ' systo ' does not exist or cannot be
  used.' '.
ENDIF.

CLEAR I_T_RETURN.

CALL FUNCTION 'BAPI_SRCSYSTEM_GETDETAIL'
  EXPORTING
    SOURCESYSTEM = systo
  IMPORTING
    * TEXTLONG =
    * DBSYSTEM =
    * BASECLASS =
    * PORTNAME =
    * SYSTEMTYPE =
    RETURN = I_S_RETURN
    * METHOD =
    * PROGRAM =
    * GWHOST =
    * GWREGION =
    * HOST =

IF I_S_RETURN IS NOT INITIAL.
  MESSAGE E051(rsar) WITH 'Source system ' systo ' does not exist or cannot be
  used.' '.
ENDIF.

CLEAR I_T_RETURN.

* Get InfoPackages in selected process chain

SELECT * FROM rspcchain INTO TABLE l_t_chain WHERE chain_id = copypc
  AND objvers = 'M' AND type = 'LOADING'.

LOOP AT l_t_chain INTO l_h_chain.
  l_infopackage = l_h_chain-variante.
  CALL FUNCTION 'BAPI_IPAK_GETDETAIL'
    EXPORTING
      INFOPACKAGE = l_infopackage
    IMPORTING
      DETAILS = I_S_DETAILS
      SCHEDULING_PARAMS = I_S_SCHEDULING_PARAMS
      DESTINATIONS = I_S_DESTINATIONS
      FILE_PARAMS = I_S_FILE_PARAMS
      HIE_PARAMS = I_S_HIE_PARAMS
      TABLES
        SELECTI ONS = I_T_SELECTI ONS
        INFOCUBES = I_T_INFOCUBES
        THIRD_PARTY_PARAMS = I_T_THIRD_PARTY_PARAMS
        RETURN = I_T_RETURN.

* Check whether InfoPackage is connected to selected source system

IF I_S_DETAIL-LOGSYS EQ sysfrom.

* Change InfoPackage description

*
CONCATENATE prefix I_S_DTLAI LS- TEXT INTO I_text.
I_text = I_text(60).
MOVE I_text TO I_S_DTLAI LS- TEXT.

* Change InfoPackage Source System

MOVE systo TO I_S_DTLAI LS- LOGSYS.

CALL FUNCTION 'BAPI__IPAK_CREATE'
EXPORTING
   DETAILS            = I_S_DETAILS
   SCHEDULING_PARAMS  = I_S_SCHEDULING_PARAMS
   DESTINATIONS       = I_S_DESTINATIONS
   FILE_PARAMS        = I_S_FILE_PARAMS
   HIE_PARAMS         = I_S_HIE_PARAMS
IMPORTING
   INFOPACKAGE        = t_infopackage
TABLES
   SELECTI_PARAMS     = I_T_SELECTI_PARAMS
   INFOCUBES          = I_T_INFOCUBES
   THIRD_PARTY_PARAMS = I_T_THIRD_PARTY_PARAMS
RETURN             = I_T_RETURN.

l_s_ipaks-ur_logdpid   = l_infopackage.
l_s_ipaks-ziel_logdpid = t_infopackage.
APPEND l_s_ipaks TO l_t_ipaks.
clear t_infopackage.
clear l_infopackage.
clear I_text.
END IF.
ENDLOOP.

* create new chain as copy of the given chain

CREATE OBJECT l_r_chain
EXPORTING
   i_copy_from     = copypc
   i_new           = 'X'
EXCEPTI ONS
   aborted_by_user = 1
   not uniqe        = 2
   OTHERS           = 3.
IF sy-subrc <> 0.
   MESSAGE ID sy-mgd TYPE 'E' NUMBER sy-mgn
   WITH sy-mgv1 sy-mgv2 sy-mgv3 sy-mgv4.
END F.

* replace the trigger process

READ TABLE l_r_chain->p_t_chain INTO l_s_trigger
   WITH KEY type = 'TRIGGER'.
CALL METHOD l_r_chain->replace_process
EXPORTING
   i_s_old         = l_s_trigger
EXCEPTI ONS
   aborted_by_user = 1
   OTHERS           = 2.
IF sy-subrc <> 0.
   MESSAGE ID sy-mgd TYPE 'E' NUMBER sy-mgn
   WITH sy-mgv1 sy-mgv2 sy-mgv3 sy-mgv4.
END F.
* ---- Save the modified version ----

CALL METHOD l_r_chain->save.
IF sy-subrc <> 0.
  MESSAGE ID sy-msgid TYPE 'E' NUMBER sy-msgno
    WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
ENDIF.

* ---- there is no method to exchange processes without popup ----
* so we need to do it on the DB. For this first free
* the chain object

l_chain = l_r_chain->p_chain_id.
FREE l_r_chain.

* ---- Get the new chain from DB and replace iaks ----

SELECT * FROM rspcchain INTO TABLE l_t_chain WHERE chain_id = l_chain
AND objvers = 'M'.
LOOP AT l_t_iaks INTO l_s_iaks.
  READ TABLE l_t_chain WITH KEY type = 'LOADING' variante = l_s_iaks-ur_logdpid
    INTO l_s_chain.
  l_tabix = sy-tabix.
  IF sy-subrc <> 0.
    MESSAGE i051(rsar) WITH 'Package ' l_s_iaks-ur_logdpid ' not in given
    chain' ''.
  ELSE.
    l_s_chain-variante = l_s_iaks-ziel_logdpid.
    MODIFY l_t_chain FROM l_s_chain INDEX l_tabix.
  ENDIF.
ENDLOOP.

* ---- Replace DB ----

DELETE FROM rspcchain WHERE chain_i_d = l_chain
AND objvers = 'M'.
INSERT rspcchain FROM TABLE l_t_chain.
COMMIT WORK.

* ---- Re-instantiate the chain from DB ----

CREATE OBJECT l_r_chain
EXPORTING
  i_chain = l_chain
EXCEPTIONS
  aborted_by_user = 1
  not_unique = 2
  OTHERS = 3.
IF sy-subrc <> 0.
  MESSAGE ID sy-msgid TYPE 'E' NUMBER sy-msgno
    WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
ENDIF.

* ---- Activate it ----

CALL METHOD l_r_chain->activate
EXPORTING
  i_no_pl an = 'X' "if you want to put the chain directly into batch, set a
space here
EXCEPTIONS
  errors = 1
  warnings = 2
  OTHERS = 3.
IF sy-subrc <> 0.
  MESSAGE ID sy-msgid TYPE 'E' NUMBER sy-msgno
    WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
ENDIF.
MESSAGE s051(rsar) WITH 'Generation of process chain ' l_chain ' has been succesful' .
FREE l_r_chain.

*        Show it       *

CALL FUNCTION 'RSPC_CHAIN_MAINTAIN'
EXPORTING
i_chain         = l_chain
EXCEPTI ONS
  + _ INTERNAL_ERROR  = 1
  + _ ABORTED_BY_US E R = 2
  + _ OTHERS          = 3
IF sy-subrc <> 0.
  MESSAGE ID sy-msgid TYPE 'E' NUMBER sy-msgno
      WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
ENDIF.

Appendix 5: Code for a Characteristic Value Variable on the 0CLIENT InfoObject

The following code has to be written in include ZXRSRU01 of SAP enhancement RSR00001:

CASE i_vnam. " <== This CASE statement is for ALL variables
  WHEN '<TECHNICAL NAME OF CLIENT-VARIABLE>'.
    * get client from user name
      DATA: l_client TYPE /bic/oiclient.
      REFRESH e_t_range.
      SELECT SINGLE client FROM zuser_client INTO (l_client)
          WHERE uname = sy-uname.
      IF sy-subrc = 0.
        * user found
          l_s_range-sign     = 'I'.'.
          l_s_range-opt      = 'EQ'.'.
          l_s_range-low      = l_client.
          l_s_range-high     = 'I'.'.
          APPEND l_s_range TO e_t_range.
      ENDIF.
  END CASE.

Appendix 6: Code for a Characteristic Value Variable on the 0CLIENT InfoObject when Using SET / GET Parameters

This code serves as a template if you are using SET / GET parameters (user master data):

CASE i_vnam. " <== This CASE statement is for ALL variables
  WHEN '<TECHNICAL NAME OF CLIENT-VARIABLE>'.
    * get client from user name
      DATA: l_client TYPE /bic/oiclient.
      REFRESH e_t_range.
      GET PARAMETER ID 'CLIENT' FIELD l_client.
      IF sy-subrc = 0.
        * user found
          l_s_range-sign     = 'I'.'.
          l_s_range-opt      = 'EQ'.'.
          l_s_range-low      = l_client.
          l_s_range-high     = 'I'.'.
          APPEND l_s_range TO e_t_range.
      ENDIF.
  END CASE.
Appendix 7: Reference to SAP Service Marketplace Notes

The following SAP Notes were considered when producing this document:

- 315094: Recommendations for authorizations in BW reporting
- 637692: Authorization check when selecting filter
- 667389: INPUT_INVALID in CL_RSD_DTA upon filter value selection/F4 help
- 699317: Routines / formulas are changed alternately
- 729227: New variables for characteristic 0REQUID