ENABLING SAAS SOLUTIONS WITH MULTITENANT BI BEST PRACTICES
4 Introduction
5 Deployment Models
5 Shared Database
5 Multiple Database with Shared Semantic Layer
5 Multiple Database with Distinct Semantic Layer
6 Data Security
6 Shared-Database Deployment Model
8 Cross-Tenant Data Security
8 Model Involving Multiple Databases with a Shared Semantic Layer
9 User-Level Data Security
9 Cross-Tenant Data Security
9 Model Involving Multiple Databases with Distinct Semantic Layer
9 User-Level Data Security
9 Cross-Tenant Data Security
9 SAP Crystal Reports with Direct Database Access
10 SAP BusinessObjects BI Platform
10 User Groups
10 BI Content Organization
10 Model Involving a Shared Database with Shared Semantic Layer
10 Model Involving Multiple Databases with Shared Semantic Layer
10 Model Involving Multiple Databases with Distinct Semantic Layer
11 Access Control Setting
12 Tenant Onboarding
12 Procedure for a Shared-Database Deployment Model
12 Procedure for Multiple Databases with Shared Semantic Layer
12 Procedure in Multiple Databases with Distinct Semantic Layer
13 User Provisioning
13 User Name Uniqueness Consideration
14 BI Content Change Management
14 Centrally Controlled Content
14 Stock Content Modified by Tenant
14 Centrally Managed Content with Tenant Customization
15 Bursting of Personalized BI Documents
15 Bursting Mode
16 Profile Value
17 Usage Management
18 Conclusion
Business intelligence (BI) software is increasingly a critical factor in enterprise IT architecture and a prerequisite for most enterprise application sales. For vendors providing software as a service (SaaS), the goal is to provide a full solution that seamlessly integrates operational and analytic functionality. The idea is to provide customers with full visibility and insight into their business, while making sure their data is kept separate and secure from other customers.

The software industry has recognized multitenancy as a key deployment characteristic enabling an SaaS solution to support many customers simultaneously, in a cost-effective manner. While the definition of multitenancy may vary among analysts, the basic premise is that resources and infrastructure are shared among tenants while, at the same time, virtual isolation is provided across the tenant base. Generally, the higher up the technology stack at which the resources are shared, the greater the potential for cost saving (both hardware and operational). For instance, multitenancy capability, implemented at the application layer, generally offers more cost savings than running separate operating system instances (physical or virtualization).

The SAP® BusinessObjects™ Business Intelligence (BI) platform offers a range of flexible features to enable the use of a single deployment to serve multiple tenants – in other words, it is a multitenant application. This document describes the best practices to successfully implement the SAP BusinessObjects BI platform in a multitenant environment. Please note that the product name SAP BusinessObjects BI platform has replaced the former name SAP BusinessObjects Enterprise, which may continue to appear on product documentation.

To get the most use and benefit from this document, you must have a basic knowledge of the SAP BusinessObjects BI platform, including the overall architecture, core security model, usage of semantic layers (“universes”) and BI documents. For a better understanding of the architecture, please review chapter 2 of the SAP BusinessObjects Enterprise Administrator’s Guide (http://help.sap.com/businessobject/product_guides/boexir31SP3/en/xt31_sp3_bip_admin_en.pdf).

Please follow the order of the chapters in this document to make sure you fully understand the core concepts before drilling into specific details.
One key factor that determines the best practice to deploy BI in a multitenant environment is the storage of the business data on which the BI tools need to operate. This section outlines three deployment models commonly adopted by SaaS partners. Other specific deployments can be considered an extension or combination of these three models. In subsequent chapters, best-practice advice is provided in reference to these three models.

Shared Database

In this deployment model, data for all tenants is stored in a single database. Semantic layers built on top of the database are shared among the tenants (see Figure 1).

Issues to consider when using the shared-database model:
- This model maximizes sharing of resources and typically has the best economy of scale.
- Operation and maintenance of BI content in the SAP BusinessObjects BI platform is simpler.
- Database security can be more complicated than other models.
- Variation of data structure among tenants is limited.

With this model, it is easy to support aggregated metrics for peer comparison.

Multiple Database with Shared Semantic Layer

In this deployment model, each tenant’s data is stored in a separate database and, because the schema of each tenant’s database is virtually the same, a single universe can be shared among the tenants (see Figure 2).

Issues to consider when using a model involving multiple databases with a shared semantic layer:
- This model has stronger data isolation than the shared-database model and may sometimes be required for regulatory reasons.
- The database design and security model may be simpler than the shared-database model.
- Operation and maintenance of BI content in the SAP BusinessObjects BI platform is still simple, similar to that of the shared-database model.
- This model shares fewer resources than the shared-database model and is typically more costly.

With this model, it is more difficult to support aggregated metrics for peer comparisons.

Multiple Database with Distinct Semantic Layer

In this deployment model, each tenant’s data is stored in a separate database, and the schemas of the tenants may have variations that call for different universes among tenants (see Figure 3). Often the database schemas and the universes are customizations or extensions of a generic model.

Issues to consider when using a model involving multiple databases with a distinct semantic layer:
- This model has strong data isolation, as in the model involving multiple databases with a shared semantic layer.
- The data security model is simplest among the three models.
- This model offers more flexibility in database and universe design.
- The operation and maintenance cost is the highest among the three models.
- This model has the least sharing of resources and is typically the most costly.
- It is challenging to support aggregated metrics for peer comparisons.
Data security is the most important consideration in the design of a multitenant system. It amounts to designing the database, universes, and BI documents in such a way that a tenant’s data would not be visible to other tenants. Furthermore, when a user within a tenant consumes a piece of BI content, only the information related to that user is shown. The technique to achieve user-level data filtering is also influenced by the deployment model.

Shared-Database Deployment Model

User-level data security: The following approaches can be adopted to achieve data filtering on a per-user basis.

- **Database-level security** – In this approach, we leverage the specific database’s inherent row-level security functionality. The database is designed and implemented in such a way that row-level security is automatically applied for each user’s logon information. Refer to your database vendor’s documentation on how you can implement the database’s row-level feature. In the SAP BusinessObjects BI platform, database-level security can be achieved by using the enable database credentials feature of the user management function in the SAP BusinessObjects BI platform (aka DBUSER and DBPASS variables). This feature associates a database user name–password pair with each user account in the platform (see Figure 4). When the universe connection is set to use the authentication mode Use BusinessObjects credential mapping, it will take the DBUSER and DBPASS values from the user information and pass them to the database at runtime (when document is viewed or scheduled).

Refer to the SAP BusinessObjects Enterprise XI3.1 Universe Designer User Guide, chapter 2, “Setting Universe Parameters,” for further details on using the database credential associated with the user record in the SAP BusinessObjects BI platform.

This approach has strong data protection because enforcement is performed at the lowest layer. It requires more sophisticated security inside the database, but simplifies the design of the universe.

It is advisable that the DBUSER/DBPASS values on the user record in the SAP BusinessObjects BI platform be managed by a software development kit (SDK) application that is invoked as part of the user management routine of the containing business application. Doing this ensures that it can be automated and kept in sync with the database. Refer to the SAP BusinessObjects Enterprise Java SDK Developer Guide for an overview of the SAP BusinessObjects BI platform SDK, as well as the SAP BusinessObjects Enterprise Java SDK Reference for more information on the IUsers interface. Specifically, the application can set the value of the DBUSER/DBPASS attributes in the user record in statements similar to the following:

```java
user.setProfileString("DBUSER", dbuserValue);
user.addSecondaryCredential("DBPASS", dbpassValue);
```

- **Dynamic SQL filtering via entitlement table** – In this approach, a table in the database has a list of users and indicates the relevancy of other data to this user. Typically this table is joined to other tables when a query asks for information relevant to that user.

An example of the table is depicted in Figure 5.

In the SAP BusinessObjects BI platform, the universe must be designed to make use of the entitlement table.
This approach can keep the universe design simple, resulting in relatively low maintenance overhead. However, it does require use of an entitlement table in the database, and the user ID in the entitlement table must be the same as the user name in the SAP BusinessObjects BI platform. (See “Tenant Onboarding” in this guide for further discussion on user name management.) Furthermore, reports that need to aggregate data across multiple users within a tenant (for example, a report run by the tenant administrator) may require the use of a different universe or application of the filter individually at the universe level, instead of using the mandatory global filter.

- **Row-access restriction in universe** – In this approach, data filtering is achieved by using universe row-access restriction. (See Figure 7.) The list of users or user groups and their corresponding row-access restriction are defined inside the universe.

<table>
<thead>
<tr>
<th>UserID</th>
<th>Name</th>
<th>UserID</th>
<th>OrderID</th>
<th>OrderID</th>
<th>Date</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Adam</td>
<td>1</td>
<td>1230</td>
<td>1230</td>
<td>03/17/2010</td>
<td>4030</td>
</tr>
<tr>
<td>2</td>
<td>Brian</td>
<td>1</td>
<td>6245</td>
<td>6245</td>
<td>07/14/2010</td>
<td>500</td>
</tr>
<tr>
<td>3</td>
<td>Cary</td>
<td>2</td>
<td>4332</td>
<td>4332</td>
<td>02/04/2010</td>
<td>2764</td>
</tr>
<tr>
<td>4</td>
<td>David</td>
<td>3</td>
<td>6342</td>
<td>6342</td>
<td>11/01/2010</td>
<td>625</td>
</tr>
<tr>
<td></td>
<td>UserOrder</td>
<td>3</td>
<td>5736</td>
<td>5736</td>
<td>05/20/2010</td>
<td>746</td>
</tr>
</tbody>
</table>

User ID: UserID (entitlement information)

Figure 5: Example of the Table in the Database

and to generate the appropriate WHERE clause that restricts access based on the identity of the user. One technique is the usage of the @variable(“BOUSER”) construct, where BOUSER represents the user account name and its value is passed to the universe dynamically. For example, if “jsmith” was logged on to InfoView, then @variable(“BOUSER”) would return “jsmith.” This variable can be applied to the WHERE clause of a universe object to filter the data relevant to the current user.

Furthermore, if the same user filtering condition is applicable whenever a user accesses data via the universe, a condition object can be defined and applied as a mandatory filter at the universe level (see Figure 6). This can reduce the effort of putting the @variable(“BOUSER”) one by one in universe objects.

You will find more details about the usage of @variable in chapter 9, “Optimizing Universes,” in the XI3.1 Universe Designer User Guide, as well as details on condition objects and mandatory filter in chapter 5 of the guide, “Building Universes.”

This approach can keep the universe design simple, resulting in relatively low maintenance overhead. However, it does require use of an entitlement table in the database, and the user ID in the entitlement table must be the same as the user name in the SAP BusinessObjects BI platform. (See “Tenant Onboarding” in this guide for further discussion on user name management.) Furthermore, reports that need to aggregate data across multiple users within a tenant (for example, a report run by the tenant administrator) may require the use of a different universe or application of the filter individually at the universe level, instead of using the mandatory global filter.

- **Row-access restriction in universe** – In this approach, data filtering is achieved by using universe row-access restriction. (See Figure 7.) The list of users or user groups and their corresponding row-access restriction are defined inside the universe.

You can find more details about access restriction in chapter 10, “Managing Universes,” of the Universe Designer User Guide.
The benefit of this approach is the flexibility in defining the data access among users. For example, users with similar access requirements can be put into a user group in the SAP BusinessObjects BI platform with an access restriction applied to this group. This doesn't require the existence of an entitlement table in the database and allows adjustment of the data filtering process at the universe layer without changing the database.

The drawback of this approach, as compared to the entitlement table approach, is the maintenance of the access restriction in the universe. This process may be automated and integrated into the onboarding process using the universe designer SDK and the SAP BusinessObjects BI platform SDK.

The entitlement table approach and the row-access restriction approach both require the proper security setting inside the universe. Since this deployment model is a single universe shared among multiple tenants, the universe must be configured to disable modification by any tenant.

Cross-Tenant Data Security
While the previous section explains the technique to filter data on a per user basis, there are situations when data must be obtained and aggregated for multiple or all users within a tenant. In this case, care must be taken to ensure that the data retrieved does not include data from other tenants. Two techniques may be used to achieve this:

- **Row-access restriction** – In general, the discussion of this technique in the previous section (under user-level data filtering) also applies here. The main difference is that the definition of the restriction typically includes a WHERE clause that uses the tenant ID rather than the individual user ID.

- **Alternative table-access restriction** – An alternative to using row-level data filtering is to put data for different tenants into different tables. Inside the universe, access restriction of type “alternative table” can be applied to users or user groups, ensuring that the tenant-specific table is used at runtime in place of a generic table (see Figure 8).

While this approach has stronger data isolation than the row-access restriction technique, it requires more maintenance overhead on both the database side and the universe side.

**Model Involving Multiple Databases with a Shared Semantic Layer**

The major difference between this deployment model and the shared-database model is the need to switch to a specific database based on the identity of the user. This can be achieved by using a combination of the following:

- Create one universe connection object for each tenant.
- Within the universe, apply an access restriction of type Connection to the user group for the tenant and select the corresponding connection object for the access restriction. Figure 9 illustrates a combination of one universe connection with a Connection access restriction.

![Figure 8: Table-Access Restriction](image1)

![Figure 9: Universe Connection with Connection Access Restriction](image2)
User-Level Data Security
The discussion of user-level data security under the shared-database model applies to this model as well.

Cross-Tenant Data Security
This is achieved automatically by the nature of database separation.

Model Involving Multiple Databases with Distinct Semantic Layer
Compared to the model involving multiple databases with a shared semantic layer, no connection switching is required because a different universe is associated with a different connection object.

User-Level Data Security
The discussion on user-level data security under shared-database model applies to this model as well.

Cross-Tenant Data Security
This is achieved automatically through database separation.

SAP Crystal Reports® with Direct Database Access
The previous recommendations center on the usage of the universe to provide the data connectivity and data security management. In the case where SAP Crystal Reports® software with direct data access is desired, without usage of the universe, additional techniques can be employed to fulfill the data security needs.

Using BOUSER variable in SAP Crystal Reports
In the SAP Crystal Reports formula language, the function CurrentCEUserName returns the logon user name of the user viewing the report. This function can be used in the record selection formula to restrict the data retrieved, ensuring that only data relevant to the user is shown. For example, the selection formula may contain a clause like the following:

EntitlementTable.userName = CurrentCEUserName

In this approach, since the data restriction is defined in the selection formula, the report template must be secured so that end users cannot modify the selection formula. This can be achieved by revoking the "modify" capability from the end users.

Using DBUSER/DBPASS in SAP Crystal Reports
Typically, when using SAP Crystal Reports with direct database connection, an explicit database logon credential must be provided. The system may prompt the end user to enter the database credential at viewing or scheduling time, or the credential may be configured at the report property and shared among users so that they are not prompted.

If prompting end users to enter database credentials is undesirable, yet different users need to connect to the database with different logon credentials to leverage the data security defined inside the database, the DBUSER/DBPASS variable may be used. As explained in the previous section, the DBUSER/DBPASS variable is associated with the SAP BusinessObjects BI platform logon account. In this case, instead of consuming the variable in a universe connection, it is consumed by SAP Crystal Reports. To enable usage, the following configuration should be made. Note that this technique is only applicable to viewing reports with live data, but not for scheduling reports.

– At the database configuration of the SAP Crystal Reports software, enter the literal "dbuser" as the database user name, and the literal "dbpass" as the password (see Figure 10).
– Add a command line option –use-SecondarySSO to the SAP Crystal Reports cache server in the central management console (see Figure 11).

Report bursting – Reports with personalized data view, showing only the data a user is allowed to see, can be generated using the publication functionality. See “Bursting of Personalized BI Documents” in this guide for more information.

Figure 10: Database Configuration for an SAP® Crystal Reports® Document

Figure 11: SAP® Crystal Reports® Cache Server Configuration
This section describes the best practice in managing the security on the entities inside the SAP BusinessObjects BI platform.

User Groups

It is advisable that tenants each have their own user groups in the SAP BusinessObjects BI platform. This will help simplify the authorization management in the shared SAP BusinessObjects BI platform and will make applying it to the semantic layer more effective and to manage.

Furthermore, to implement a delegated administration model, such as sub-administrators per tenant (for users and content), you can create subgroups within each tenant group to separate these special users from the rest. These subgroups can then have different security privileges applied such as application rights and semantic layer restriction sets. For instance, for the tenant named “ABC,” a user group ABC_administrators can be created to refer to the tenant-level administrator, and a user group ABC_bi_consumers can be created to refer to the nonprivileged BI content consumer. Figure 12 depicts the user-group structure of a fictional system that illustrates the idea.

BI Content Organization

The following outlines how to organize BI content in various deployment models.

Model Involving a Shared Database with Shared Semantic Layer

Regular users of the system typically consume the service via BI documents (SAP BusinessObjects Web Intelligence® software, SAP Crystal Reports, and so on). To simplify content security and management, it is advisable that each tenant has its own home in the public folder tree. This folder is typically created at the top level of the folder tree and named after the tenant.

In the case where a report document is shared by all tenants, this report can be stored within a hidden central folder and copied to the tenant’s folder. This can be automated via scripts and is typically part of the tenant onboarding process, which is discussed further in “Tenant Onboarding” in this guide.

The universe and connection objects can be stored in the same central location since they are shared among the tenants in this scenario (and are not configurable).

Model Involving Multiple Databases with Shared Semantic Layer

For BI content management, the same advice applies: each tenant should have its own public folder for storing its BI documents.

The universes can be stored in a central folder as they are shared among the tenants in this scenario.

For universe connections, each tenant has its own connection object; however, in the XI3.1 version of the software, connection objects cannot be organized into folder tree. The visibility of the connection object by unrelated tenants can be restricted by placing the access control list onto the connection object itself.

Model Involving Multiple Databases with Distinct Semantic Layer

Both the BI documents and the universe objects should be stored in tenant-specific folders.
Access Control Setting

Access control on BI content objects
- The access control setting specifies who can carry out operation on specific content. While powerful access control technology in the SAP BusinessObjects BI platform allows many variations and complexity, the security setting can be simplified and easily managed if user-group and folder structures are adopted.

Additionally, Access Level should be used to encapsulate the details of the allowed operations. For more details on using access level, you can refer to the XI3.1 SAP BusinessObjects Enterprise Administrator’s Guide, chapter 18, “Setting Rights,” and the “Rights Appendix” of the document for the definition of each of the rights defined in the system.

Figure 13 depicts a typical access control list of the document folder belonging to a fictional tenant ABC.

Access control on user and user-group objects
- In a multitenant environment, it may not be desirable for the users of a tenant to know about the existence of other users or user groups belonging to other tenants. This comes down to managing the “view” right a user has onto the user objects and user-group objects. However, it should also be noted that user objects inherit rights from two sources:
  - The top-level system folder that contains all user objects
  - The parent user-group object that the user object belongs to

Similarly, user-group objects inherit rights from both a top-level system folder as well as from parent user groups. To achieve the result that users in a tenant see all users belonging to that tenant but no other users, the practice illustrated in the example below can be applied:
- Assume a user group – ABC_users – is created to represent users for the tenant ABC.
- For the access control list on the group object ABC_users, add the group ABC_users as a security principal with view access. This will allow all users belonging to the user group to view the existence of ABC_users, as well as its subgroup and contained users (see Figure 14).
- To make the group ABC_users and its users invisible to other users outside the group, grant the view right capability onto the top-level all users folder and top level all groups folder to the everyone group, within the scope of this object only. This allows everyone to see the existence of the top-level folder, but view right will not be granted to the contained users and groups. The view right to the top-level folder is still required because some UI workflow needs to start the navigation from this folder (see Figure 15).
This section describes the procedures that are typically taken when a tenant is added to the system. As a rule of thumb, the procedure is standardized across all tenants in a deployment so it can be repeated and automated effectively. Automation can be achieved by using the SAP BusinessObjects BI platform SDK and the universe designer SDK.

**Procedure for a Shared-Database Deployment Model**

The following steps are typical in systems adopting the shared-database deployment model:
1. Create a top-level public folder for the tenant to store BI content. Prefix the name of the folder by the tenant identifier.
2. Create a user group to represent all users of the tenant. Prefix the name of the user group by the tenant identifier.
3. Create subgroups under the main tenant user group to represent personas with different access privileges (see the section titled “User Groups” in this guide).
4. Set the access control list on the tenant document folder and the user group accordingly (see the section titled “Access Control Setting” in this guide).
5. Copy the stock BI documents into the tenant document folder. This can be automated via the SAP BusinessObjects BI platform SDK.
6. Create user accounts in SAP BusinessObjects BI platform for each tenant user (see “User Provisioning” below for further discussion).
7. If universe access restriction is used to achieve user data filtering (see “Shared-Database Deployment Model” in this guide), add the access restriction definition for the user groups or users into the universe.

**Procedure for Multiple Databases with Shared Semantic Layer**

The following steps are typical in a system adopting a deployment model involving multiple databases with shared semantic layer:
1. Create a top-level public folder for the tenant to store BI content. Prefix the name of the folder by the tenant identifier.
2. Create a user group to represent all users of the tenant. Prefix the name of the user group by the tenant identifier.
3. Create subgroups under the main tenant user group to represent personas with different access privileges (see “User Groups” in this guide).
4. Create a universe connection for the tenant’s database.
5. Set the access control list on the tenant document folder and the user group accordingly (see “Access Control Setting” in this guide). Also grant the tenant user group the rights to the connection object created in step 4.
6. In the shared universe, add a connection access restriction for the tenant user group so that all tenant users will use the connection created in step 4.
7. Copy the stock BI documents into the tenant document folder.
8. Create user accounts in the SAP BusinessObjects BI platform for the users in the tenant (see “User Provisioning” below for further information).
9. If universe access restriction is used to achieve user data filtering (see “Shared-Database Deployment Model” in this guide), add the access restriction definition for the user groups or users into the universe.

**Procedure in Multiple Databases with Distinct Semantic Layer**

The following steps are typical in a system adopting a deployment model involving multiple databases with distinct semantic layer:
1. Create a top-level public folder for the tenant to store BI content. Prefix the name of the folder by the tenant identifier.
2. Create a user group to represent all users of the tenant. Prefix the name of the user group by the tenant identifier.
3. Create subgroups under the main tenant user group to represent personas with different access privileges (see “User Groups” in this guide).
4. Create a universe connection for the tenant’s database.
5. Set the access control list on the tenant document folder and the user group accordingly (see “Access Control Setting” in this guide). Also grant the tenant user group the rights to the connection object created in step 4.
6. In the shared universe, add a connection access restriction for the tenant user group so that all tenant users will use the connection created in step 4.
7. Copy the stock BI documents into the tenant document folder.
8. Create user accounts in the SAP BusinessObjects BI platform for the users in the tenant (see “User Provisioning” below for further information).
9. If universe access restriction is used to achieve user data filtering (see “Shared-Database Deployment Model” in this guide), add the access restriction definition for the user groups or users into the universe.
If users and user groups for the tenant are stored in an LDAP or Active Directory system, at tenant onboarding time, the corresponding tenant user groups must be mapped into the SAP BusinessObjects BI platform. Once the group is mapped to the SAP BusinessObjects BI platform, users inside the AD/LDAP group will be imported into the software as part of the group mapping process.

User Name Uniqueness Consideration

User names in the SAP BusinessObjects BI platform must be unique. However, in some cases the containing business application supports nonunique user names across different tenants. In these situations, construct the user name in the SAP BusinessObjects BI platform using a combination of the tenant name as well as the user name within a tenant, for example, ABC_AdamSmith where ABC is the name of the tenant.

However, if the user name must be passed through the universe to the database, the account name in the SAP BusinessObjects BI platform using a combination of the tenant name as well as the user name within a tenant, for example, ABC_AdamSmith where ABC is the name of the tenant.

User Provisioning

User accounts can be created in the SAP BusinessObjects BI platform for the end user to make use of the BI service. In the case where BI is a subset of functionalities within a bigger business application, it is typical that the containing application is the primary source of the user definition. Two approaches are generally applicable in such environments:

- Creating user accounts with the SAP BusinessObjects BI platform authentication type using the SDK
  When user accounts are created in the main business application, the tools and processes used to manage these accounts can be enhanced to make use of the SDK to create the corresponding user accounts.

  Furthermore, if the end users start with the business application first before they consume the BI services, the two systems (the contained business system and SAP BusinessObjects BI platform) can be configured to support single sign-on. This will provide a better user experience and will remove the need to synchronize passwords in the SAP BusinessObjects BI platform. One specific technique to achieve this is Trusted Authentication. You can find more details about trusted authentication in the XI3.1 SAP BusinessObjects Enterprise Administrator’s Guide, chapter 12.

- Using external authentication
  Another approach to simplify user provision is to make use of authentication systems or directory services outside the SAP BusinessObjects BI platform; that is, to create accounts of authentication type Lightweight Directory Access Protocol (LDAP) or Active Directory (AD).

If users and user groups for the tenant are stored in an LDAP or Active Directory system, at tenant onboarding time, the corresponding tenant user groups must be mapped into the SAP BusinessObjects BI platform. Once the group is mapped to the SAP BusinessObjects BI platform, users inside the AD/LDAP group will be imported into the software as part of the group mapping process.

User Name Uniqueness Consideration

User names in the SAP BusinessObjects BI platform must be unique. However, in some cases the containing business application supports nonunique user names across different tenants.

In these situations, construct the user name in the SAP BusinessObjects BI platform using a combination of the tenant name as well as the user name within a tenant, for example, ABC_AdamSmith where ABC is the name of the tenant.

However, if the user name must be passed through the universe to the database, the account name in the SAP BusinessObjects BI platform user cannot match the user name in the database, as this may not be prefixed by the tenant name. In this case, it is recommended that DBUSER/DBPASS be leveraged instead of BOUSER. DBUSER variable does not need to be unique and can store the plain user name as that used inside the database, without the tenant prefix.
After the tenants have been onboarded, it is possible the BI content consumed may undergo changes. This section describes the issues to consider in managing the change effectively.

In general, the change management of BI document and universe objects depends on the ownership of the content and the degree of change allowed by the tenant. This may be broadly classified into the following types:

**Centrally Controlled Content**

In this scenario, the content is owned by the central administrator; an individual tenant is not supposed to modify it. This is the simplest scenario, and the integrity of the content can be obtained easily. The downside is the lack of flexibility of change by the tenant.

Change management practice in this scenario can be summarized as:

- Disallow modification of the content by tenants
- Modify the content and test the change in test environment
- Copy the changed content into tenant-specific folder in the test system, if each tenant is supposed to have a duplicate copy of the BI content
- Promote the modified content from the test system to the production system. This can be achieved through the import wizard, BusinessObjects Intelligence Application Resource (BIAR) command line tool, or lifecycle manager.

**Stock Content Modified by Tenant**

In this scenario, a set of stock content is provided by the central administrator. The tenant can then modify the content as appropriate. Once modified, the content is considered to be separate from the standard stock content, so modification of the stock content cannot be merged with modifications made by the tenant. This applies more often to BI documents than to universes.

The advantage of this approach is that the current tenant can start with useful BI content while gaining the flexibility of making changes. The drawback is that major changes in the database or universe may impact the BI content, making it more challenging to contain the impact centrally.

Change management practices in this scenario can be summarized as:

- Copy the stock content to the tenant-specific folder.
- Allow modification to the content in the tenant-specific folder by the tenant.
- When change is needed on the stock content, modify the content and test the change in a test environment.
- Promote the modified stock content from the test system to the production system.
- As an option, copy the updated stock content to the tenant-specific folder if the tenant-specific copy of the content has not been modified (this may be achieved by checking the last modification time).

**Centrally Managed Content with Tenant Customization**

In this scenario, the content is created and managed by a central administrator. However, customizability is built into the content so that each tenant can consume the content with variation, without actually changing the design of the content. The advantage of this approach is that changes can still be managed centrally while variation and custom experience is provided. The drawback of this approach is that the design of the BI content tends to be more sophisticated, involving more maintenance, while the level of flexibility is not as high as an offspring copy owned by the tenant completely.

The following outlines some of the techniques that may be adopted to achieve this kind of managed variety.

- **Hidden object in universe**
  The database may have extra columns that are used by some but not all tenants. The corresponding universe object can be hidden from the irrelevant tenants by applying access restriction of type object access. You can find more information on access restriction in the XI3.1 Universe Designer User Guide, chapter 10.

- **Customized universe by API**
  For more modification to the universe, a copy can be made from the master universe and modified by using the universe designer application program interface (API). API is used instead of manual modification so that the modification can be reapplied programatically when there is a change in the core universe. For more information about the universe designer API, see the XI3.1 Universe Designer API Reference. (Note that the universe designer API in the XI3.1 version of the software only runs on Microsoft Windows).

- **Parameterized report**
  BI reports can be designed to take parameter at runtime, with elements in the report (for example, filter condition or formatting of text) adjusted based on the parameter value. Variables and formula language can be used in the report to achieve more sophisticated adjustments.
Sometimes the query against the database may take too long, resulting in a poor end-user experience. In these cases, the BI documents can be preprocessed via the scheduling functionality in the SAP BusinessObjects BI platform, in order to retain a snapshot of the data.

For BI documents that convey the same information to all users within the tenant, scheduling the BI document will suffice. However, if different users need to see different data from the same report template, it is advisable to use the publication functionality in the SAP BusinessObjects BI platform. For more information about the publication functionality, please refer to the SAP BusinessObjects Enterprise XI3.1 Publisher’s Guide.

The SAP BusinessObjects BI platform provides options to fine-tune the behavior to match the database security design or performance characteristics of the system. The following summarizes the key considerations in choosing the most appropriate options.

**Bursting Mode**

One key decision needs to be made when configuration publication is the "bursting mode," because this affects how many queries would be sent to the database:

- **One database fetch per recipient**
  In this bursting mode, a report processing job is created for each recipient behind the scenes, and each job will become a separate query to the database.

- **One database fetch for all recipients**
  In this bursting mode, a single query is sent to the database to retrieve data for all users. Then filtering for different users is performed inside the report, and personalized copies of the master report are created for distinct users.

The benefit of this approach is the ability to leverage the data security already defined inside the database or universe. This can be used for all three techniques of user-level data security described in the "Data Security" section of this guide (database logon credential, entitlement table, universe restriction).

This bursting mode is easier to configure than the other bursting modes. However, it results in sending many queries – one for each user – to the database. The total processing time as such may take longer than the other bursting modes. The actual difference depends on the performance characteristics of the database.

To make use of this mode, the slice of data that each user is supposed to see should be stored as "profile value" in the SAP BusinessObjects BI platform. See the following section for further discussion on profile value.

In this bursting model, the BI report in the publication is processed in the user context of the scheduler of the publication. It is important that the user account used to schedule the publication will retrieve all relevant data for all users without unwanted
filtering at the semantic layer or database level. For example, a “tenant administrator” account may be used.

In the shared-database deployment model, the administrator can run a single publication for the entire system or one publication per tenant. However, in multiple database deployment models (either shared or distinct semantic layer), each tenant will need to run its own publication (the processing of a publication only involves one database).

■ One database fetch for a batch of recipients
In SAP BusinessObjects BI platform, version XI3.1, this bursting model is supported by SAP Crystal Reports, but not SAP BusinessObjects Web Intelligence and SAP BusinessObjects Desktop Intelligence™ software.

This bursting model is similar to the “one database fetch for all recipients” mode. For example, profile values need to be created, and the publication job needs to be run with the administrator account. However, instead of running a single job and sending a single query to be database, the system will break up a publication job into multiple batches automatically. Each child job will fetch data for a collection of users but not all users. This can reduce the potential memory consumption in a single report with a big row set as in the “one database fetch for all recipients” mode. This method usually also offers a higher level of parallel processing than the “one database fetch for all recipients” mode.

Profile Value

To use the “one database fetch for all recipients” or “one database fetch for batch of recipients” bursting mode, the filtering values for the users should be stored in “profile value” in the SAP BusinessObjects BI platform. Profile offers a flexible mechanism to store the filtering value for the users – that is, user ID, cost center code, country, department name, and so on. Profile value can also be assigned to the user group once (for example, department name for all users in a department), and the users in that group will inherit the setting dynamically.

Profile values can be entered at central management console, under the “Profiles” section. This can also be set using the SAP BusinessObjects BI platform SDK. You can find more information about managing profile in the XI3.1 SAP BusinessObjects Enterprise Administrator’s Guide, chapter 27, “Managing Profile.” For information on using the API to manage profile value, please refer to the section “IProfileValue” in the XI3.1 SAP BusinessObjects Enterprise Java SDK Reference.
When the central administrator wants to measure the usage of the system by the users of each tenant (for example, for billing purposes), the administrator may leverage the data stored in the auditing database of the SAP BusinessObjects BI platform.

Data in the auditing database is comprised of “audit events,” which represent activities users have carried out. Each audit event stores the ID of the user carrying out the activity, together with other key information about the activity. Each event type has a list of event details defined. One of these event details is user-group information (identified by event detail type 42), which conveys a list of user-group names to which the user carrying out the activity belongs. This user-group audit event detail can be used to extract or summarize audit information for a tenant.

Note: When a user belongs to multiple user groups, the user-group event detail will contain a string concatenating all the groups to which the user belongs, such as “Everyone|TenantABC.” While this string may need further processing to extract the tenant name, the string itself in the raw form can still be used to separate users in one tenant from users in another tenant.
As explained in this document, the SAP BusinessObjects BI platform has a range of functionalities to enable BI in a multi-tenant environment. Furthermore, various techniques may be employed, depending on the needs and operating characteristics of the solution. When choosing the most suitable approach, the following factors should be considered:

- How much isolation of data is required by the business?
- What is the data volume of the tenant, and what level of sharing will produce the best economy of scale?
- To what extent can the database and BI content be standardized across tenants? How much flexibility is required?
- How often and extensive is the database schema or BI content changed during the life of the solution? What is the expected maintenance effort?
- What is the performance characteristic of the database query? Should the BI document be viewed with live data, or with cached data using scheduling/publication?
- Are development resources available to leverage the SAP BusinessObjects BI platform SDK to automate some operation?

This document provides a broad-based analysis of the various approaches and best practices to address these questions, as well as the relative pros and cons of each approach. Once the premise of these best practices is understood, readers are encouraged to review other product documentation for further technical details.