SIEBEL eBusiness 7

SIEBEL PERSONALIZATION ADMINISTRATION GUIDE

VERSION 7.5, REV. A

12-FAUN8W

MARCH 2003

Siebel Systems, Inc., 2207 Bridgepointe Parkway, San Mateo, CA 94404

Contents

Introduction

Chapter 1. About Siebel Personalization

Chapter 2. User Profiles

Chapter 3. Visibility

Chapter 4. Content Targeting

Chapter 5. Run-time Events

Chapter 6. Testing

Appendix A. Siebel Query Language

Index

Introduction

This guide describes how to use Siebel Personalization for customizing enterprise-wide interactions with users, whether customers, partners, or employees.

Siebel Personalization is a core infrastructure component of Siebel eBusiness Applications and is fully integrated into the Siebel architecture. Implementation does not require Siebel Tools configuration.

This book will be useful primarily to people whose title or job description matches one of the following:

Call Center Administrators Persons responsible for setting up and maintaining a call center. Duties include designing and managing Computer Telephony Integration (CTI), SmartScripts, and message broadcasts.

Database Administrators Persons who administer the database system, including data loading; system monitoring, backup, and recovery; space allocation and sizing; and user account management.

Marketing Administrators Persons responsible for setting up and maintaining a marketing department. Duties include designing and managing campaigns, product marketing information, and product distribution lists.

Siebel Application Administrators Persons responsible for planning, setting up, and maintaining Siebel applications.

Siebel Application Developers Persons who plan, implement, and configure Siebel applications, possibly adding new functionality.

Siebel System Administrators Persons responsible for the whole system, including installing, maintaining, and upgrading Siebel applications.

How This Guide Is Organized

This book is divided into chapters that describe particular features of Siebel Personalization and how to use them. An appendix describes the functions and operators used in Siebel Query Language.

Revision History

Siebel Personalization Administration Guide, Version 7.5, Rev. A

March 2003 Bookshelf

Book Version: Rev A

Table 1. Changes Made in Rev. A for March 2003 Bookshelf

Topic	Revision
"Dynamic User Profile Attributes"	Added ActiveViewName to list of attributes.
"Creating Actions"	Added new content to Business Service Context discussion in table.

Additional changes

- Removed screenshots and updated text in procedures to reflect current standards.

About Siebel Personalization 1

Siebel Personalization provides an integrated multichannel personalization platform for customizing enterprise-wide interactions with customers, partners, and employees. It is part of the core infrastructure of Siebel eBusiness Applications and is fully integrated into the Siebel architecture.

Siebel Personalization Components

Siebel Personalization has two main components:

- **Personalization rules.** Controls visibility and content in user interface elements such as applets and views. Personalization rules are evaluated at run time and use the available information about the user, such as his or her profile and any transaction information. Rules can control the visibility of views and applets or control the contents displayed in an applet.

  Personalization rules are created and managed in the Personalization Administration screens.

  For more information on controlling the visibility of views and applets, see Chapter 3, "Visibility." For more information on creating personalization rules, see Chapter 4, "Content Targeting."

- **Run-time events.** Allows applications to change in real time based on user inputs. Run-time events are triggered in response to user actions. Business managers can configure the actions in response to any event.

  Run-time events and the actions that trigger them are created and managed in the Runtime Events Administration screen. For more information on run-time events, see Chapter 5, "Run-time Events."

Using Siebel Personalization

In addition, Siebel Personalization provides:

- **End-user layout customization**. Users can customize the layout directly from portal-style pages. Applets can be moved, hidden, and minimized using the Edit Layout button.
  - Applets can be moved to various positions in the view using the Edit Layout button. This allows end users to express their preference as to where applets should appear.
  - Applets can be hidden using the Hide button, then shown again using the Edit Layout button (assuming that the administrator allows this to happen). This allows end users to show only those applets that are most important to them.
  - Applets can be minimized using the Collapse button so that they appear on the view, but only the title bar appears. This allows end users to keep information that is infrequently used hidden until it is needed.
- **Personalization on behalf of another user.** Users have access to their own profile when they log in, but they can also read the profile of other users with whom they are interacting.

  For example, a call center agent can load the profile of the user with whom he or she is speaking to trigger personalization rules on that user's behalf. This allows personalization to work consistently across all channels.
- **Personalization Business Rules Designer.** The Personalization Business Rules Designer allows business managers to implement personalization rules without learning complex programming languages.
  - The Rules Designer uses Siebel Query Language to construct expressions. The if-then rule structure facilitates rule writing.
  - The template-based approach allows business managers to write rules by filling out a template wizard with the appropriate key words, while not getting in the way of expert users who just want to type rules in the expression fields.

  - The Rules Designer provides choices of rule elements based on the context of the rule. The context comes from the place where the Rules Designer is invoked: action, applet, rule, and so on.

  For more information, see "Personalization Business Rules Designer" on page 55.

- **Advanced testing environment**. Creating personalization rules is by nature a dynamic process. How the system responds to a user depends on who that user is, and what actions they perform while using the system. Testing personalization rules before deploying them on a live environment allows an administrator to assess whether the rules are having the desired effect.

  - The administrator can test the effects of content targeting rules, and personalization events and actions in employee, customer, and partner applications.

  - The administrator can load the profiles of "dummy" users. These users can simulate the effect of personalization rules on hypothetical user profiles.

  For more information, see "Using Test Mode to Test Personalization Rules" on page 88.

Siebel Personalization Within the Siebel Architecture

Siebel Personalization is installed automatically as part of the Siebel object manager. It is administered in run time, providing maximum control without the necessity of recompiling the repository. This allows business managers to anticipate customer needs based on their actions and react to them in real time.

In the Siebel architecture, there are objects that define the user interface, business objects that define the business rules, a data manager for moving data in and out of the database, and the database itself.

Siebel Personalization forms a layer between the user interface and the business objects. Figure 1 shows the Siebel architecture, which consists of Run-time administration, data, and business objects from the Siebel repository.

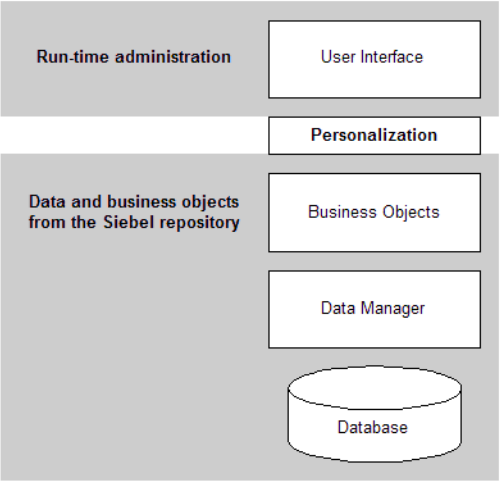


Figure 1. Siebel Architecture

Overview of Using Siebel Personalization

Use the following process to start using Siebel Personalization. Each of these activities is covered in more detail in this document.

NOTE: Siebel Personalization is active in all applications by default.

- Make sure that all persons engaged in personalization administration are assigned a responsibility in the Siebel system that allows them to see all Personalization Administration and Runtime Events Administration views. This is especially important if you have configured multiple organizations in the Siebel system.

  If you delegate any authority to channel partners for administering personalization, make sure your LDAP server is configured to expect these logins.

- Analyze business requirements for personalized behavior. Consider requirements for employees and channel partners as well as customers.
- Create persistent user profile attributes as necessary to track the user information needed for personalization.
- Create rules to control the visibility of views, applets, and content based on user profile attributes and content attributes.
- Register events that you want to monitor in the Events and Events Aliases view.
- Configure actions to be triggered when events of interest occur.

NOTE: When a user is not specifically logged into the Siebel application, the user actually has a session as a guest user. Using this session, an anonymous user’s experience can be personalized. For example, using personalization rules an anonymous user can be configured to see a different home page from a logged in user.

Roles and Responsibilities

There are two main roles in working with Siebel personalization:

- **Business manager or analyst.** Analyzes and defines personalization requirements that meet business needs in the most cost-effective manner. Requests personalization behavior based on the experience of employees, customers, or channel partners.
- **Personalization administrator.** Translates the business requirements into personalization rules, rule sets, events, and actions. Also responsible for reviewing and maintaining the implementation.

  The personalization administrator must have the responsibility associated with the Siebel administrator to have the necessary access to perform these duties.

Personalization Terminology

Some of the terms used in this section are explained below:

Personalization The ability to provide specific information and application functionality based on known or inferred characteristics of a user. Decisions are made in real time about what to show or not show to a user based on the profile built up over time.

Content All the visible components of a display, including data, text, images, and structural elements.

Content Targeting The ability to display information dynamically to users based on their interests and behavior, either explicitly stated or implicitly observed.

Profiling The ability to collect information about users in order to deliver the most appropriate and effective content. Profiling is accomplished using registration information, by explicit polling, and by real-time observation of user behavior.

Rules-Based Filtering The ability to show users information based on rules defined by a system administrator or marketing manager.

Personalization and Content Targeting

Siebel Personalization makes it possible to deliver a customized message to a large number of employees, customers, and channel partners across all channels. It is important to drive personalization decisions from a business perspective, that is, to understand the business issues you are trying to address before implementing personalization.

Siebel Personalization brings together your detailed understanding of your content and your users. Figure 2 illustrates that by using content and users you define content attributes and build the user profile. Both are used to create rules.

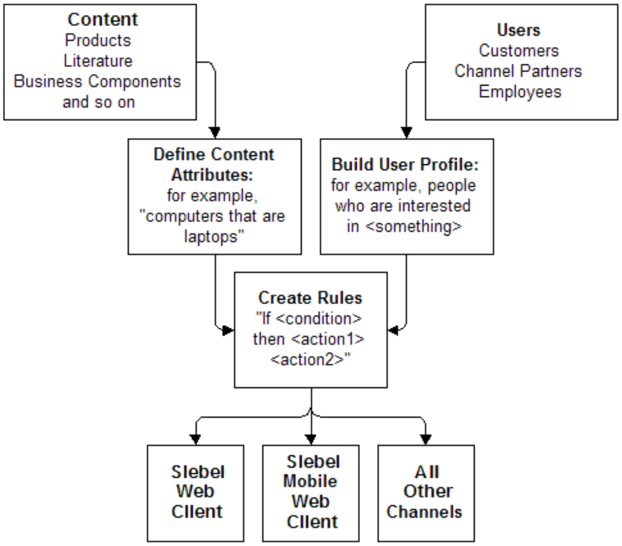


Figure 2. Content Targeting

The key to implementing personalization is to understand the target audience, whether customer, channel partner, or employee. When the target audience is identified, then the business imperative for that audience must be clearly defined.

You should consider the following:

- What views should be shown or hidden, depending on the user interacting with the system?
- Should certain applets be shown or hidden?
- What opportunities are there to show customized content to users based on their profiles?
- What events take place that should be tracked? Alternatively, are there business requirements for tracking certain events?
- Can benefits be derived from delivering personalized content to employees as well as customers and channel partners?

Once a personalization system is in place, then the business manager and personalization administrator must develop an ongoing strategy to monitor and implement changes in business requirements as well as evaluate the effectiveness of the current personalization plan.

Usage Scenarios

Users experience Siebel Personalization as content targeted to them, such as on the home page. The personalized home page displays opportunities, expenses, a calendar, and other content specific to the user.

The following sections provide more examples of using Siebel Personalization.

Real-Time Product Recommendations

A company sells thousands of products in many areas. A typical customer is likely to be interested in a small subset of those products. The company wants to recommend only those products that are of interest to a particular customer. The company also wants to make sure that the products shown are in stock and are profitable.

Use the following steps to implement this:

- Create or borrow the recommended products user interface (a recommended items applet already exists in Siebel eSales).
- Write personalization rules to target content based on user profile attributes or other transaction data, such as past orders. For example, customers can be shown products based on the industry they specified when they registered.
- Write personalization rules to recommend products based on customer behavior. For example, when a customer adds a product to his or her Shopping Cart, the ProductInCart persistent or dynamic user profile attribute can be set. Products defined in the Product Administration screen as being related to that product can be displayed to the customer for cross-sell and upsell opportunities.

  The following rule:

  ```
  EXISTS([Related Product] = GetProfileAttr("ProductInCart"))
  ```

  shows products that are related to products in the Siebel eSales Shopping Cart.
- Create a business service to access external data or check for business constraints. For example, an ERP system can be used to determine whether a product is in stock and how well it has been selling recently.
- Write personalization rules to access the business service and to show or hide products based on availability and profitability.

When customers enter the company’s Web site, they see products appropriate to their industry or other interests. After adding products to the Shopping Cart, new products are recommended that complement the ones they have already selected. The customers are more likely to buy products that are useful and readily apparent.

Personalization on Behalf of Another User

A company wants to provide the same level of personalized service through all its channels, using the same set of personalization rules.

The company has Siebel eSales Web site that it uses to sell its products. On this Web site the company recommends products to its customers.

The company also runs a call center. When a customer calls the company's call center, the company wants its call center agents to recommend products based on the profile of the caller rather than the profiles of the agents.

The company also wants to recommend the same products to the customer as those recommended on the Web site.

Use the following steps to implement this:

- Create or borrow the recommended products user interface (a recommended items applet already exists in Siebel eSales).
- Create personalization rules to target content based on the user profiles for both the logged-in user (the call center agent) and the secondary user (caller). These are called the "Me" and "You" profiles, respectively.
- Load the secondary user profile ("You") when a customer calls.

When the customer calls the company, the call center agent knows the customer's interests, concerns, service request history, and past purchases. The customer gets personalized attention and a more efficient response from the call center agent. The agent can recommend useful products to the customer for cross-sell and upsell opportunities.

Integration With Third-Party Personalization Engines

A company wants to prevent its customers from switching to a competitor's product by giving them special offers.

The company analyzes historical data and discovers a trend that is related to customer defection. Based on this analysis, the company creates a new model that uses relevant customer data to predict the probability of a customer switching to a competitor's product.

The model might be created using a third-party personalization engine. Several such models exist for the pharmaceutical industry.

The company wants to monitor customer actions in real time while running the model. Based on the model's predictions, the company wants to make a special offer to likely defectors to prevent them from switching.

Use the following steps to implement this:

- Analyze historical data and create a new model that accesses the relevant data about customers and predicts the probability of their defection.
- Create a business service that integrates Siebel with the third-party software and runs the predictive model.
- Write personalization rules that trigger the business service in response to customer actions. These run-time personalization events are passed to the third-party engine through the business service interface.
- Write personalization rules to deliver a special offer to a customer based on results from the model.

When the customer enters the Web site, he or she might fill out a survey or just browse the site. A special offer appears in a new browser window. Because the offer is targeted specifically to the customer, he or she takes the time to look at it instead of exiting the Web site.

User Profiles 2

User profiles consist of attributes that you want to track for each user, so time invested in analyzing your audience is well spent.

As you determine your business requirements for personalization, review the standard user profile attributes available and decide if you need to create additional attributes.

User Profile Attributes

User profile attributes can include personal information about the user, such as name, address, email address, and hobbies, or a variety of extended information, such as company, industry, position, responsibilities, items purchased, information requested, or service provided.

User profile attributes are configurable in Siebel Tools. Once modeled, they are available across all channels, whether the user contacts your organization through your Web site, your call center, or any other channel managed through your Siebel system.

About Managing User Profile Attributes

User profile attributes can be either dynamic or persistent, and can be managed in a variety of ways across many channels:

- Self–administration of a user's own attributes, for example, updating his or her mailing address. These attributes make up the "Me" user profile.
- By a call center agent for customers or others contacting your organization by phone, for example, a service request.

  The profile of the call center agent profile is the "Me" profile. The profile of the customer for whom the agent is managing the attributes is called the "You" profile. The "You" profile is represent by the current contact in the customer dashboard.
- By a field or sales representative administering customer or contact profile attributes remotely on a site visit and then synchronizing the changes with a corporate database.

User profile attributes must be specified exactly as they appear in the business component, and are case sensitive. For example, if `Channel` is the correct user profile attribute, it must not be spelled `channel`.

Persistent User Profile Attributes

The persistent user profile consists of persistent user profile attributes, both individual and organizational. These attributes are stored in the Siebel database.

Persistent profile attributes exist in the base tables or extension tables of the S_PARTY table. They can be either person-based or organization-based. They appear as fields in the Personalization Profile business component. They can also be present in any S_PARTY-based business component, but the Siebel Personalization Engine loads only those defined in the Personalization Profile business component. All organization-based attribute names start with an `Org.` prefix.

Basic information such as name and email address should be stored as persistent attributes. Your business needs will determine what other attributes are important to store from one session to another. In general, the most frequently used attributes are made part of the persistent profile.

Values for persistent attributes are loaded in the session when a user logs in. Persistent attributes are retrieved using `GetProfileAttr()`. For more information on `GetProfileAttr()`, see "Profile Functions" on page 111.

When persistent attributes are modified by a user and the session is updated, either directly or by the user's actions while logged in, the modified values are not saved to the database until the session is terminated. This is because the profile attributes are cached. Caching profile attributes allows access to them to be extremely fast when building pages, while providing persistence between sessions.

When persistent attributes are modified in an administration screen and saved, for example by a call center agent using the Profile view under the User Profile Preferences screen, the changes are saved to the database at that time.

NOTE: GetProfileAttr only works for fields explicitly defined in the Personalization Profile business component. GetProfileAttr does not work with system fields, which are not explicitly defined in the business component; it returns a NULL value for them.

An exception is the Id system field. This particular system field is available to GetProfileAttr even though it is not in the Personalization Profile business component.

Dynamic User Profile Attributes

Dynamic attributes are created at run time and are not based on any business component fields. They are session specific.

Dynamic attributes are set and modified by actions the user takes while navigating the system, and are never written to the database. They allow real-time modification of the user experience. They are related to the state of the user, as determined when he or she entered the application.

The login process can also automatically enter dynamic attributes. The following dynamic profile attributes are set by the system at run time:

Dynamic Profile Attribute	Possible Values
IsStandaloneWebClient	“TRUE” or “FALSE”
IsAnonymous	“TRUE” or “FALSE”
ApplicationName	Name of the application
ActiveViewName	Name of the active view

Dynamic attributes are retrieved the same way as persistent attributes, using `GetProfileAttr()`. If the attribute is not set (or does not exist) for a given user, it will return nothing, but will not give an error message.

Dynamic profile attributes can be different for each user. When a dynamic profile attribute is set, it exists for that user only. This works because of how dynamic attributes are set and retrieved: no errors occur if an administrator references dynamic attributes that do not exist. If a dynamic profile attribute does not exist and the user sets it, a new one is created.

NOTE: The ActiveViewName profile attribute is set after the user has been navigated to the view.

The following are examples of the usage of dynamic profile attributes:

- `GetProfileAttr(“AttrDoesNotExist”)` returns NULL.
- `SetProfileAttr(“NewAttribute”,”value”)` creates a new dynamic profile attribute.

Loading a Second User Profile

In Web channel interactions, only one party is involved. However, in certain other channel interactions, for example Siebel Call Center, there are multiple parties involved, for example, call center and customer agent. In this situation, it is necessary to load a second user profile. The second profile is used if a call center agent wants to personalize the content shown on the screen based upon the caller's profile instead of the agent's own profile.

About Loading a Second User Profile

The following describes instances of loading a second user profile when the customer profile is open and when the customer is not using CTI.

- If the customer dashboard is open when the caller contacts the Call Center, the second profile is automatically loaded when the dashboard is set with the user's credentials.

NOTE: The second profile will be loaded if the Call Center agent executes a find for the customer contact and uses the Set Dashboard button to launch the dashboard.

- If the customer is not using CTI, the customer can write a script or business service to call the function LoadUserAttributes with the contact ID, and use this script to load the second user profile.

NOTE: LoadUserAttributes is exposed for the application object. This function takes only one argument: the row-id of the employee or contract.

Accessing the Second User Profile

The second user profile can be accessed with personalization rules by using a you.indentifier. The following are examples of accessing the second user profile:

```
GetProfileAttr (“me.first name”)
```

returns the first name of logged in user

```
GetProfileAttr (“you.firstname”)
```

returns the first name of the second user

Using LoadUserProfile

The following VB example shows a method that loads a user profile into the session. The function is exposed on the Siebel Application Object.

```
Function LoadUserProfile As Integer

sArgs = "0-10N07"

TheApplication.InvokeMethod "LoadUserAttributes", sArgs

End Function
```

This function has only one parameter: the row-id of the person whose profile needs to be loaded.

If this function is called with empty parameters, it unloads the loaded user profile. For example:

```
Function LoadUserProfile As Integer

sArgs = ""

TheApplication.InvokeMethod "LoadUserAttributes", sArgs

End Function
```

Adding New Persistent User Profile Attributes

You can add new persistent user profile attributes by adding columns to the S_PARTY table or one of its extension tables, and then exposing these columns as business component fields. New profile attributes are available after the repository file is recompiled.

About the Personalization Profiles Business Component

All persistent user profile attributes are stored in the Personalization Profile business component, which has two sets of fields:

- **Person-related.** All person-related fields are based on the S_PARTY table or its extension tables and joins that point to a person's attributes. Examples of such extension tables are S_CONTACT, S_CONTACT_X, and S_USER.

  The best way to define a new field is to find a similar one in the person-based business components, such as User and Employee, and use it as a prototype.

- **Organization-related.** All organization-related fields start with `Org.` and are based on the S_PARTY table or its extension tables and joins that point to an organization's attributes. Examples of such extension tables are S_ADDR_ORG, S_ORG_EXT, and S_POSTN.

  The best way to define a new field is to find a similar one in the organization-based business components, such as Account and Internal Division, and use it as a prototype.

Querying the Personalization Profile Business Component

The Personalization Profile business component is queried in a certain way, so it is important to use the `Org.` prefix for organization-related fields:

- The business component is positioned to the record corresponding to the person whose attributes are being read.
- All person-related fields are read.

- Either the Primary Account Id Internal or Primary Division Id Internal field (whichever is not empty) is used as an Id field to query for the organization-related fields.

  Because both person and organization records are stored in the S_PARTY table and its extensions we can use the same business component to query for these fields.

- All fields with names `Org.*` are read.

When persistent user profile attributes are saved, for example, when exiting the application, a similar process happens.

Restrictions

The following restrictions apply when working with the Personalization Profile business component:

- All MVG- and join-based fields (except extension table–based fields) are read-only. Flag them accordingly in Siebel Tools.
- If adding an MVG field, only its primary value is made visible to the end user.
- Do not use any inner joins in the Personalization Profile business component, as it may prevent the business component from being queried correctly.
- Do not modify the definitions of the following fields, because they are used to implement Siebel Personalization functionality:
  - Primary Account Id
  - Primary Account Id Internal
  - Primary Division Id
  - Primary Division Id Internal
  - Primary Position Id

- Do not modify the definitions of the following joins, as they are used to implement Siebel Personalization functionality:
  - Primary Account
  - Primary Division
  - Primary Position
- For performance reasons, flag fields in Siebel Tools as read-only whenever possible. This will avoid an attempt to save the record if the corresponding attribute is changed.

For information on working with business components and tables, see *Siebel Tools Reference*.

Performance Considerations

Figure 3 shows that in Siebel Personalization, a profile attribute consists of a person component and an organization component.

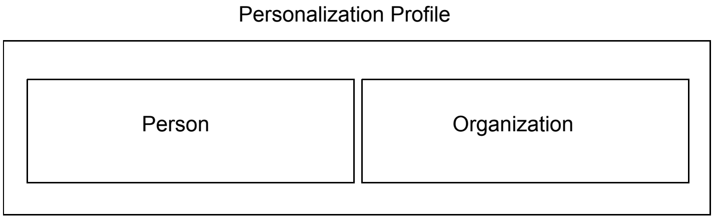


Figure 3. Personalization Profile

When you implement action sets and events associated with the Personalization profile attributes, you should consider the following:

- It takes two queries to load profile attributes and two queries to save them. One query is for the person, and the second query is for the organization. These attributes are loaded the first time Siebel Personalization accesses a persistent profile attribute.
- If only a *person's* persistent attributes are set, the system will not load and save the *organization* component of the Personalization profile and vice versa.
- Using only dynamic attributes will save two queries on start up and two on exiting as the persistent profile would not be loaded or saved.
- If the user does not modify a persistent profile attribute during a session, then two queries are saved because nothing is saved to the database.

Working with Multiple Value Profile Attributes

Profile attributes with multiple values are stored as MVGs. You configure them in Siebel Tools as ordinary MVG (Multiple Value Group) fields on the personalization profile business component. For more information, refer to "Adding New Persistent User Profile Attributes" on page 31.

Using MVG to Improve Searches

Siebel Personalization provides a hook for entering dynamic search specifications on applets and business components. Administrators can use these dynamic search specifications to target the content to the user based on their profile information. To extend the profile, Personalization now supports MVG attributes, which can be used to track information on customers that cannot be captured in single value fields. These attributes could be hobbies, interests, and so on.

Profile Attributes in Siebel 7.5

In Siebel 7, MVG profile attributes were not supported. Administrators could still use comma delimited lists as a substitute for MVGs, for example, profile attributes of states where there are Siebel offices might have a value of CA,MA,GA. In Siebel 7.5, profile attributes with more than one value are supported through MVG profiles.

About the GetProfileAttrAsList Function

A new function, GetProfileAttrAsList, is introduced in Siebel 7.5. This function returns the MVG value as a list, but can also be used in the EXISTS operator to create the right expressions for matching the MVG profile attributes with content within Siebel applications.

The following describes using the GetProfileAttrAsList outside and within the EXISTS operator:

- When GetProfileAttrAsList is used outside the Exists operator, it returns a comma separated list of the MVG value.

  For example, if you are using a MVG called State that has the values CA, MA, GA, and CA is primary, then GetProfileAttrAsList (State) would return CA, MA, GA.

- Within the EXISTS operator, the GetProfileAttrAsList function returns the value of the profile attributes in the format expected by the operator. For example, a typical usage of the EXISTS operator in this scenario would be EXISTS ([Targeted States] = GetProfileAttrAsList("State")). This would do a many-to-many match of the MVG Business Component Field Targeted State against the MVG profile attribute State.

- Semantics of GetProfileAttr also should be modified for MVG profile attributes. So if GetProfileAttr function is called for an MVG, it returns only the primary value of the MVG.

  For example, an MVG called State, with the values CA, MA, GA, where CA is primary, GetProfileAttr (State) would return CA.

- For matching content with MVG profile attribute, EXISTS operator would be used.

  There are three potential cases when using EXISTS with a profile attribute:

  - MVG profile attribute and MVG Business component field:

    Expression: EXISTS ([Targeted State] = GetProfileAttrAsList("State"))

    This expression tests whether any of the state**s** in the MVG profile attribute, State, exist in the BC MVG field Targeted State.

  - MVG profile attribute and single value business component field:

    Expression: EXISTS ([State] = GetProfileAttrAsList("State"))

    This expression tests whether any of the states in the MVG profile match states in the Targeted State single value BC field.

- Single value profile attribute and MVG Business component field:

  Expression: EXISTS ([Targeted State] = GetProfileAttr("State"))

  This expression tests whether single value state in the user profile match states in the MVG Targeted State BC field.

Usage Scenarios

The following are usage scenarios using the EXISTS operator.

- Searching for an MVG field in a business component against an MVG profile attribute.

  For example, a bookseller site wants to recommend books from all the categories that interest a user. A book can belong to multiple categories, that is book ABC, can belong to action, adventure, and history. Similarly, a user XYZ might be interested in multiple categories of books, that is, science fiction and action.

  The personalization engine will match the user XYZ to all the books that belong to the categories that the user is interested in, including book ABC. This done by matching the MVG profile field Interest with the MVG field Categories in the products business component.

  Expression: EXISTS ([Categories] = GetProfileAttrAsList("Interests"))

- Searching for an MVG field in a business component against a single value profile attribute.

  For example, a product is targeted to multiple states and the marketing manager wants to recommend the product to only the people who live in one of the targeted states.

  Expression: EXISTS ([Targeted State] = GetProfileAttr("Home State"))

- Searching for a single value field in a business component against an MVG profile attribute.

  For example, a product belongs to only one category, such as laptop, desktop, or palmtop, and the user is interested in buying from multiple categories, for example, palmtop and laptop.

  Expression: EXISTS ([Product Category] = GetProfileAttrAsList("Interested Categories"))

Visibility 3

This chapter describes the process of setting the visibility of views by finding the name of the view. It also describes setting the visibility of applets by finding the name of the applet, setting its visibility, and setting the number of rows displayed.

Sequence of Events Triggering Visibility

The sequence of events triggered when a user accesses a view, which determines the visibility of an applet, is shown in Figure 4.

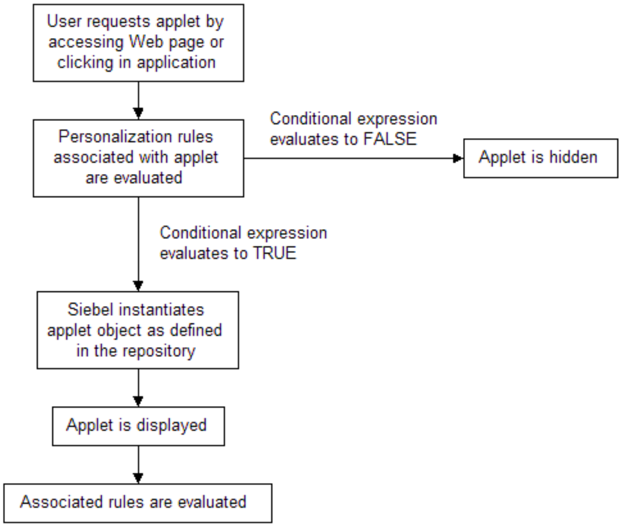

User requests applet by accessing Web page or clicking in application
Personalization rules associated with applet are evaluated
Conditional expression evaluates to FALSE
Applet is hidden
Conditional expression evaluates to TRUE
Siebel instantiates applet object as defined in the repository
Applet is displayed
Associated rules are evaluated

Figure 4. Sequence of Events Determining Applet Visibility

The evaluation of visibility conditions for views and applets is done dynamically. For example, the Shopping Cart applet in Siebel eSales can be hidden initially, and then shown based on user actions in real time.

For more information on conditional expressions, see "Conditional Expressions" on page 53.

Views That Control Visibility

You control which views users see in the Views list under Personalization Administration. For example, if you wanted to hide certain views from channel partners, you would do this in the Views list. View visibility is controlled by the conditional expression: if the conditional expression evaluates to TRUE, then the view is shown.

NOTE: You control whether or not an applet is displayed to a user and what content is shown in the Applets view under Personalization Administration. For example, the Shopping Cart in Siebel eSales can be hidden when it is empty. Like view visibility, applet visibility is controlled by the conditional expression, for example, the conditional expression for Salutation Applet (eMarketing).It is important that you have the most up-to-date repository when working with views or applets personalization. The reason is that the views and applets available to you are derived from the repository of the application in which you are working, which may not be the same as the server repository.

Visibility Rules and View Visibility

Only dynamic and persistent profile attributes can be used to write visibility rules. However, dynamic profile attributes that drive the visibility of views and applets can be created in response to events that are based on other data, such as transactions and external data.

View visibility is also affected by responsibility. Siebel Personalization can only be used to control the visibility of views and applets that are accessible to a user. For example, you cannot write personalization rules to show administration views to users without administrator responsibility.

For more information on responsibilities, see *Applications Administration Guide*.

Setting the Visibility of Views

The steps in setting view visibility are the following:

- Finding the name of the view
- Creating a conditional expression that determines whether the view is shown

Finding the Name of a View

Before you can set the visibility of a view, you must know its name. There are three ways to do this, depending on what kind of application is used:

- For employee applications, such as Siebel Call Center, navigate to the view and choose Help > About View. This displays information about the view, its applets, and its business components.
- For customer applications, navigate to the view, right-click, and then choose View Source. In the `<script>` section of the source file, the name of the view is shown after `SWEView:` in each instance of `action`.

  This method also works for employee applications.
- In Siebel Tools, choose the Screen object, query for the screen containing the view whose name you wish to know, and then select the Screen View object. This displays the views in the Name field.

  This method works for all applications.

Setting the Visibility of a View

Visibility of views is set in the Views list under Personalization Administration.

To set the visibility of a view

1. Navigate to Site Map > Personalization Administration > Views.
2. Query for the desired view.
3. If that view is not available, create a new record.
4. Click the icon in the Name field, query for the desired view, then click OK.
5. Fill in the remaining fields.

Field	Description
Start Date	The date when the conditional expression starts to be evaluated. If Start Date is blank, the conditional expression is evaluated continuously or until the End Date is reached.
End Date	The date after which the conditional expression is not evaluated. If End Date is blank, the conditional expression is evaluated continuously after the Start Date. If both Start and End Date are blank, the conditional expression is always evaluated.
Conditional Expression	An expression that causes the view to display when the expression evaluates to TRUE, and not to display when it evaluates to FALSE. Enter the expression as text or click the icon to use the Personalization Business Rules Designer. For information on conditional expressions, see “Conditional Expressions” on page 53.

Setting the Visibility of Applets

The steps in setting applet visibility are the following:

- Find the name of the applet.
- Create a conditional expression that determines whether the applet is shown.

Finding the Name of an Applet

Before you can set the visibility of an applet, you must know its name.

To find the name of an applet

1. Find the name of the view in which the applet appears (see "Finding the Name of a View" on page 42).
2. In Siebel Tools, select the View object.
3. Query for the name of the view in which the applet appears.
4. Expand the View object, and then select View Web Template.
5. Expand the View Web Template object, and then select View Web Template Item. The applets are listed under View Web Template Items.

Setting the Visibility of an Applet

Visibility of applets is set in the Applets view under Personalization Administration.

To set the visibility of an applet

1. Navigate to Site Map > Personalization Administration > Applets.
2. Query for the desired applet.
3. If that applet is not available, create a new record in the Applets list.
4. Click the icon in the Name field.
5. The Pick Applet dialog box appears.
6. Query for the desired applet, and then click OK.
7. Fill in the remaining fields as follows.

Field	Description
Start Date	The date after which the conditional expression is evaluated. If Start Date is blank, the conditional expression is evaluated continuously or until the End Date is reached.
End Date	The date after which the conditional expression is not evaluated. If End Date is blank, the conditional expression is evaluated continuously after the Start Date. If both Start and End Date are blank, the conditional expression is always evaluated.
Conditional Expression	An expression that causes the applet to display when the expression evaluates to TRUE, and not to display when it evaluates to FALSE. Enter the expression as text or click on the button to use the Expression Editor. Leaving the Expression field blank is equivalent to an expression that is always TRUE. For information on conditional expressions, see “Conditional Expressions” on page 53.

Setting the Number of Rows Displayed in an Applet

The number of rows displayed in applets is set in the [SWE] section of the configuration (.cfg) file for the application, for example:

```
[SWE]

NumberOfListRows = 7
```

To set the number of rows displayed in an applet

Change the value of the `NumberOfListRows` parameter in the `[SWE]` section of the application configuration file to the number of rows you want to see displayed.

Content Targeting 4

This chapter describes how to target content by using expressions. It also describes how to create rule sets and associate rule sets with applets. Salutation applets are also described because they use inclusion expressions of rules to display a text message to the user.

Content Targeting Steps

There are four steps to content targeting:

- **Content tagging.** Business managers tag content with attributes that describe for whom the content is most appropriate. For example, a company's products might be tagged by industry focus or value classification.
- **User profiling.** Users enter information about themselves to be used in evaluating business rules. Each piece of information is called a profile attribute. For example, a profile attribute could keep track of the industries in which a customer is interested.

  For more information on profile attributes, see Chapter 2, "User Profiles."
- **Business rules administration.** Business managers create rules that govern which content is shown to which users. These rules are written using an "if-then" syntax and allow business managers to make changes to the business logic during run time. Rules can be written to examine tagged content or the actions of users in the system.
- **Association with applets.** Business rules are associated with Siebel applets at run time, and only the content matching the rules is shown to the appropriate user.

About Controlling the Content

Controlling the content shown to users involves the following actions:

- Creating rule sets to control the content shown in applets.
- Creating the individual rules needed to accomplish the objectives of the rule sets.
- Associating the rule sets with applets and setting the sequence in which they are to be evaluated. Remember that rule sets can be shared by multiple applets.
- Creating a conditional expression for each rule set associated with an applet that prevents the rule set from being evaluated whenever there is no content available.

  The rule set conditional expression can also create complex Boolean logic within the rule set. Rules can be turned on and off based on the values of profile attributes.

Rule Sets and Rules

You create both rule sets and rules in the Rule Sets view. The relationship of rules and rule sets to applets and views is shown in Figure 5 on page 50.

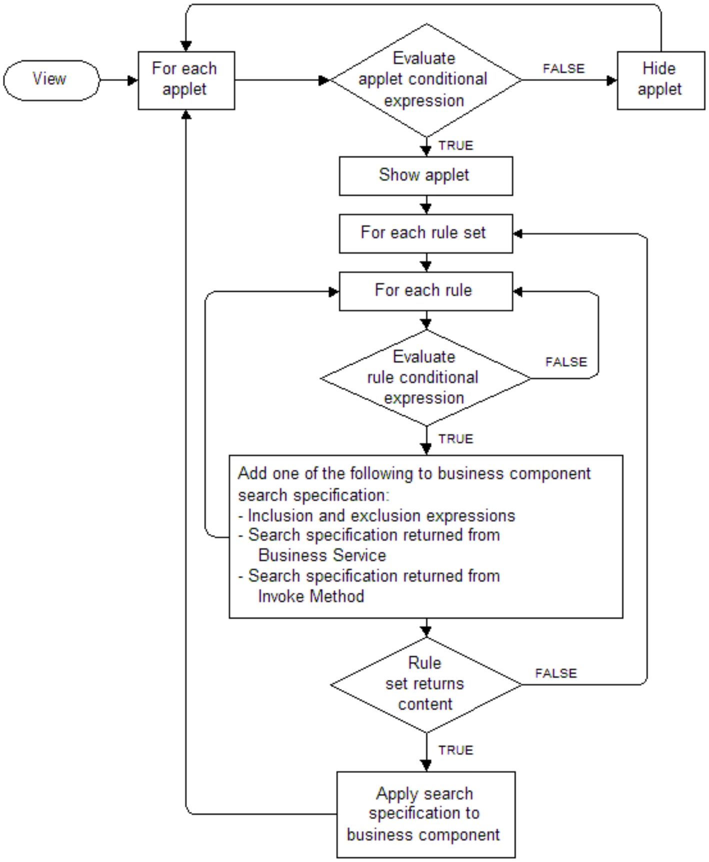

View
For each applet
Evaluate applet conditional expression
FALSE
Hide applet
TRUE
Show applet
For each rule set
For each rule
Evaluate rule conditional expression
FALSE
TRUE
Add one of the following to business component search specification:
- Inclusion and exclusion expressions
- Search specification returned from Business Service
- Search specification returned from Invoke Method
Rule set returns content
FALSE
TRUE
Apply search specification to business component

Figure 5. Relationship of Rules and Rule Sets to Applets and Views

Rule Set Evaluation

Rule sets are evaluated in the order of the sequence numbers you enter.

If a rule set is evaluated and no content or records are returned, or if the conditional expression associated with the rule set evaluates to FALSE, the next rule set is evaluated. Processing continues until a rule set returns content. As soon as a rule set returns content, processing stops. Subsequent rule sets in the sequence are not evaluated.

You can also use conditional expressions to set up Boolean logic (IF, AND, OR) among the rule sets associated with a view or an applet to create a more complex flow of evaluation.

The results of the evaluation of all the rules in the set are returned to the underlying business component associated with the applet in the form of a search expression. This search expression controls the content displayed in the applet.

Limiting the number of rule sets is good practice. This is because each rule set is a query built by adding all the rules together. The rules contribute to the final query, but do not cause a query by themselves. Therefore, you can create as many rules as you like, but create rules sets carefully.

The sequence you set for rule set evaluation is important. If the system must evaluate many rule sets before getting one that returns content, performance can be affected noticeably.

One way of managing the overhead on executing rule sets is to add a conditional expression to the rule set that evaluates to FALSE when the rule set is unlikely to return any records. This causes the rule set to be skipped whenever it is unlikely to find content.

For example, rule set 1 is based on the user’s age as calculated from their birthday and rule sets 2 and 3 are based on other information. If the user has not entered his or her birthday, then his or her age cannot be calculated. If you put a conditional expression on rule set 1 that evaluates to FALSE if the birthday attribute is blank, then the rule set is not evaluated.

Expressions

Expressions set the basic parameters for controlling the content to users. You can create expressions by entering the expression as text in the appropriate field or by using the Personalization Business Rules Designer.

Types of Expressions

There are two types of expressions:

- Conditional
- Search

Expressions are written in Siebel Query Language entered as text. You build expressions from the following basic elements:

- Functions
- Operators
- Profile attributes
- Business component fields

Creating Expressions

You can enter the expression as text directly in the expression field or launch the Personalization Business Rules Designer by clicking the icon in any selected expression field.

NOTE: The maximum size for any expression (conditional, include, exclude, and business service context) is 4096, which is limited by the database configuration.

For more information on:

- Creating expressions using the Personalization Business Rules Designer, see "Personalization Business Rules Designer" on page 55.
- Siebel Query Language, see Appendix A, "Siebel Query Language," which details the formulas and operators available.

Conditional Expressions

Conditional expressions must be written to evaluate to TRUE or FALSE. Conditional expressions trigger rules when they evaluate to TRUE.

For example:

```
GetProfileAttrAsInt(“Number of Visits”) >= 7
```

evaluates to TRUE if that profile attribute is at least 7.

Whether an applet or view is displayed is controlled by conditional expressions. What content is displayed is controlled through rule sets based on content and user profile attributes. Write your conditional expressions so that when the expression evaluates to TRUE, the view or applet is displayed or the rule set and rules are processed.

NOTE: To avoid interpretation of numbers such as telephone numbers as numeric values, enclose them in quotation marks, for example: `‘650-477-5000’`

Search Expressions

Search expressions consist of inclusion and exclusion expressions. Inclusion and exclusion expressions relate content attributes to user profile attributes in order to control what content each user sees. Each expression is part of a rule that is part of a rule set that can be linked to an applet for the purpose of controlling content.

Business components have existing visibility settings, for example, “My” visibility, “All” visibility, or “Sales Rep” visibility. There may be other search specifications configured in Siebel Tools. When a rule is evaluated at run time, its inclusion and exclusion expression are added to the business component as additional search specifications.

A rule can only return content that is within the scope of the business component’s inherent visibility or search specifications as set in Siebel Tools.

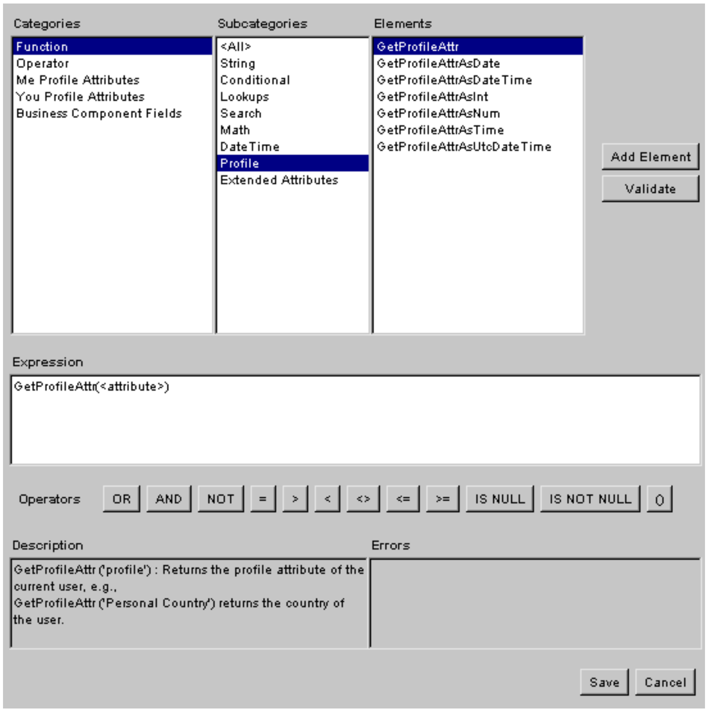
For example:

```
EXISTS([Related Product] = GetProfileAttr(“ProductInCart”))
```

shows products that are related to products in the Siebel eSales Shopping Cart.

Personalization Business Rules Designer

The Personalization Business Rules Designer (Figure 6) appears when you click the icon in any selected expression field.



Expression

GetProfileAttr(<attribute>)

Description

GetProfileAttr ('profile') : Returns the profile attribute of the current user, e.g., GetProfileAttr ('Personal Country') returns the country of the user.

Errors

Figure 6. Personalization Business Rules Designer

It contains the following:

- **Categories.** Contains the main categories available for building business rules:
  - **Function.** Siebel Query Language functions (see Appendix A, “Siebel Query Language” for more information).
  - **Operator.** Siebel Query Language operators.

- **Me Profile Attributes.** The profile attributes of the primary user, such as a Siebel eSales customer or Siebel Call Center agent.

  Valid subcategories are:

  - **ShowAsString.** GetProfileAttr is used in the expression.
  - **ShowAsNumber.** GetProfileAttrAsNum is used in the expression.
  - **ShowAsInteger.** GetProfileAttrAsInt is used in the expression.
  - **ShowAsDate.** GetProfileAttrAsDate is used in the expression.
  - **ShowAsTime.** GetProfileAttrAsTime is used in the expression.
  - **ShowAsDateTime.** GetProfileAttrAsDateTime is used in the expression.

  For more information on GetProfileAttr functions, see Appendix A, “Siebel Query Language.”

- **You Profile Attributes.** The profile attributes of the secondary user, such as a customer calling a call center agent. Uses the same subcategories as Me Profile Attributes.

- **Business Component Fields.** These depend on the context of the rule, that is, where the Rules Designer is invoked. This category will or will not have data in its elements, depending on the following contexts.

Context	Data Shown
View	None
Applet	Fields in the business component on which the applet is based
Rule set not associated with an applet	None
Rule set associated with one applet	Fields in the business component on which the applet is based
Rule set associated with more than one applet, and the applets are based on the same business component	Fields in the business component on which the applets are based

Context	Data Shown
Rule set associated with more than one applet, and the applets are based on different business components	None
Applet event	Fields in the business component on which the applet is based
Application event	None
Business component event	Business component fields
Action set not associated with an applet event or business component event	None
Action set associated with one applet event	Fields in the business component on which the applet is based
Action set associated with one business component event	Business component fields
Action set associated with more than one applet event or business component event	None

- **Subcategories.** Contains subcategories of the selected category. Its context is controlled by the Categories box and it controls the context of the Elements box.
- **Elements.** Contains all of the elements used to build business rules. Its context is controlled by the Categories and Subcategories boxes.
- **Expression.** Displays the business rule. You can type and edit in the box, as well as add elements to it.
- **Quick bar (Operators).** Provides buttons for frequently used operators.
- **Description.** Gives a description of the element selected in the Elements box.
- **Errors.** Displays errors in expressions or highlighted parts of expressions when you click Validate.

Use the following process to create business rules:

- Select an element, and then click Add Element to begin building your rule. Double-clicking an element has the same effect as Add Element. You can also type text in the Expressions box.

  The Rules Designer automatically provides the correct syntax for elements. For example, adding the Left string function displays `Left(<string>,<n>)` in the Expressions box.

- Fill in arguments in the element, and then enclose it in parentheses if necessary.
- Add other elements and complete them.
- Connect elements using the Quick bar or by typing.
- Click Validate to check your expression for errors. If part of an expression is highlighted, only the highlighted string is evaluated. Errors are displayed in the Errors box.
- Save the business rule and return to the expression field. This also validates the entire rule. If valid, the rule is saved and the Rules Designer is closed. If the rule has any errors, they are displayed and the Rules Designer is not closed.

Writing a Rule Using the Personalization Business Rules Designer

The following example shows how to write this rule:

```
EXISTS([Related Product] = GetProfileAttr(“ProductInCart”))
```

which shows products that are related to products in the Siebel eSales Shopping Cart.

To write a personalization rule using the Personalization Business Rules Designer

1. Click the icon in a selected expression field.

   The Personalization Business Rules Designer appears.

2. Choose Operator from Categories, Pattern Matching from Subcategories, and EXISTS from Elements, and then click Add Element.

   The Expression box displays `EXISTS(<condition>)`.

3. Click `<condition>` to highlight it.

4. Choose Business Component Fields from Categories, < All > from Subcategories, and Related Product from Elements, and then click Add Element.

   The Expression box displays `EXISTS([Related Product])`.

5. Place the cursor after `[Related Product]` in the Expression box, then click = in the Quick bar.

   The Expression box displays `EXISTS([Related Product]=)`.

6. Place the cursor after =.

7. Choose Function from Categories, Profile from Subcategories, GetProfileAttr from Elements, and then click Add Element.

   The Expression box displays:

   ```
   EXISTS([Related Product]= GetProfileAttr(<attribute>))
   ```

8. Click `<attribute>` to highlight it, and then type `“ProductInCart”`.

   The Expression box displays the completed rule:

```
EXISTS([Related Product] = GetProfileAttr(“ProductInCart”))
```

9 Save the rule to validate it and put it in the expression field.

Creating Rule Sets

You create rule sets in the Rule Sets view under Personalization Administration. They control the display of content. You can associate multiple rule sets with individual applets, and an individual rule set can be associated with multiple applets. Each rule set can contain multiple rules.

After you have created a rule set, you need to create the individual rules that make up the set and accomplish the objective for matching content to the user.

Types of Personalization Rules

There are three types of personalization rules:

- **Expressions.** This rule type evaluates include and exclude expressions directly. Requires a name for the rule and the rule type. This is where you create a conditional expression to control the evaluation of the rule and the include and exclude expressions that control the display of content.
- **Business Service.** Invokes a business service method. Requires a name for the rule, the rule type, a conditional expression, a business service name, a business service method, and a business service context.

  If the Business Service rule type returns a property set with either InclusionExpr or ExclusionExpr, the property set is appended to the business component search specification. Additionally, a business service can set dynamic profile attributes that can later be used to build inclusion and exclusion expressions.
- **Invoke Method.** Invokes a method on a business component. Requires a name for the rule, the invoke method context, the rule type, a conditional expression, and a method name.

Creating Rule Sets

You create rule sets in the Rule Sets list under Personalization Administration.

To create a rule set

1. Navigate to Site Map > Personalization Administration > Rule Sets.
2. Add a new record to the Rule Sets list.
3. Fill in the necessary fields.

Field	Description
Name	Name the rule set.
Active	Check the box to make the rule set active.
Start Date	Date when the rule set becomes effective.
End Date	Date when the rule set stops being effective.
Description	Provide more detail about the rule set.

Creating Rules

You create rules in the Rule Sets view under Personalization Administration.

NOTE: After modifying personalization rules, you must refresh the object manager for the rules to take effect immediately. See "Clearing and Reloading Siebel Personalization" on page 97.

To create a new rule

1. Navigate to Site Map > Personalization Administration > Rule Sets.
2. Select a rule set.
3. Add a new record to the Rules list below the Rule Sets list.

   The Rules form appears under the More Info tab below the Rules list.

4 Fill in the necessary fields.

Field	Description
Name	Choose a name that suggests the purpose of the rule.
Sequence	**Required.** Enter numbers in this field to set the order in which the rules should be evaluated. Evaluation begins with the rule with the lowest sequence number and continues until all rules are evaluated, except that -1 is evaluated last. Rules with the same sequence number are evaluated in random order. The sequence number for rules does not have the same significance as for rule sets because all the rules in a rule set are evaluated whether content is returned or not. Rule sequence can be very important, however, for the order of the text strings displayed in salutation applets.
Rule Type	Select a rule type from the drop-down menu: ■ Expressions. Evaluates inclusion and exclusion expressions directly. ■ BusService. Invokes a business service and expects a property set in return, which is passed on to the business component. The input argument to Business Service is a property set with four properties: Context, BusComp, RuleSet, and Rule. ■ Invoke Method. Invokes a method on the business component to get a string, which is the search specification. The input argument to Invoke Method is a string.
Active	Check the box to use the rule.
Conditional Expression	**Optional.** Use the conditional expression to control the evaluation of the rule, or to set up Boolean logic for evaluation of all the rules in the rule set.
Start Date	The date when the rule starts to be applied.
End Date	The date after which the rule stops being applied.
Description	Fill in this field with whatever information will help users understand the purpose of the rule.

Field	Description
Include Expression	Used with the Expressions rule type. An expression that sets parameters to include content.
Exclude Expression	Used with the Expressions rule type. An expression that sets parameters to exclude content.
Business Service Name	Name of the business service to invoke, if the conditional expression evaluates to TRUE and the rule type is BusService.
Business Service Method	Method to invoke on the business service.
Business Service Context	Parameter to pass to the business service method. While a business service can take many name/value pair parameters, you can only pass one parameter—called Context—in Siebel Personalization. An example of a business service context is `“UserType=Partner;AccountState=Gold”`.
Method Name	Method to invoke on the business component, if the conditional expression associated with the rule evaluates to TRUE and the rule type is Invoke Method.
Method Argument	Parameter to pass the business component. You can only pass one argument.

Associating Rule Sets With Applets

You use the Rule Sets list under the Applets list to associate existing rule sets with applets.

For information on creating rule sets, see "Creating Rule Sets" on page 61.

To associate rule sets with an applet

1. Navigate to Site Map > Personalization Administration > Applets.
2. Select the applet to which you want to associate a rule set.
3. Add a new record to the Rule Sets list below the Applets list.
4. Click the icon in the Name field.
5. Select a rule set, and then click Go.

   The rule set is added to the Rule Sets list.
6. Fill in the necessary fields.

Field	Description
Sequence	Choose the order in which the rule sets will be evaluated.
Start Date	The date when the conditional expression starts to be evaluated. If Start Date is blank, the conditional expression is evaluated continuously or until the End Date is reached.
End Date	The date after which the conditional expression is not evaluated. If End Date is blank, the conditional expression is evaluated continuously after the Start Date. If both Start and End Date are blank, the conditional expression is always evaluated.
Conditional Expression	Causes the rule set to be processed if the expression evaluates to TRUE.

About Salutation Applets

The salutation applet is a specialized applet that uses search specifications, defined in the inclusion expressions of rules, as message text displayed to the user. The salutation applet allows business managers to write customized messages using free-form HTML.

Siebel applications use the salutation applet to display information about the user or the session. You can set the parameters of this applet to display a variety of information taken from the user's profile or actions performed during the session.

The personalized salutation applet can greet the user by name, indicate how long it has been since the user last visited the site, list service request update notifications, and present the user with information about specific products or services that match known interests or previous behavior.

Hyperlinking Salutation Messages

Salutation messages can be localized by using the Language() function.

Salutation messages can include hyperlinks to other screens, views, and applet drilldown objects:

- **Screen.** Use the command `SWEPersonalizationGotoScreen('ScreenName');` in the hyperlink tag.

  The name of the screen must be enclosed in single quotes ('), and spaces in the name must be represented by plus signs (+).

  The following is an example of the command syntax:

  ```
  <A href=JavaScript:SWEPersonalizationGotoScreen('Accounts+
  Screen');>
  ```

- **View.** Use the command `SWEPersonalizationGotoview('ViewName');` in the hyperlink tag.

  The name of the view must be enclosed in single quotes ('), and spaces in the name must be represented by plus signs (+).

  The following is an example of the command syntax:

```
<A href=JavaScript:SWEPersonalizationGotoview('User+Profile
+View+(eApps)');>
```

You can also use this command to go to Siebel Employee Relationship Management views. The syntax is the following:

```
<A href=JavaScript:SWEPersonalizationGotoview('My+View
+Name');>
```

or

```
<A href=JavaScript:SWEPersonalizationGotoview('My+View
+Name','&SWEmyExtra1=true&SWEMyExtra2=false');>
```

Make sure there are no spaces between the two argument quotes and the comma separating them. Make sure there are no spaces in the myExtra parameters.

The ampersand for the first myExtra parameter is optional. You can also use the following syntax:

```
<A href=JavaScript:SWEPersonalizationGotoview('My+View
+Name','SWEmyExtra1=true&SWEMyExtra2=false');>
```

There is no “dup argument” checking mechanism. If your myExtra parameters contain an argument that SWE has already generated, undesired behavior might occur.

- **Applet drilldown objects.** Use the command `SWEPersonalizationDrillDown ('`*`ViewName`*`','`*`AppletName`*`','`*`FieldName`*`','`*`RowId`*`',{`*`ParentRowIds`*`});` in the hyperlink tag.
  - *`ViewName`* is the name of the view where the drilldown object is defined.
  - *`AppletName`* is the name of the applet on which the drilldown object is defined.
  - *`FieldName`* is the business component field name on which the drilldown object exists.

- `ParentRowIds` is an array of strings that contains the Row IDs of the parent records. It is optional, and is only used when drilling down in a child applet.

Names must be enclosed in single quotes (‘), and spaces in the names must be represented by plus signs (+).

The following is an example of the command syntax:

```
<A href=JavaScript:SWEPersonalizationDrillDown('Opportunity+
List+View','Opportunity+List+Applet','Name','1-45XHZ');
```

NOTE: If you wish to have two salutation applets in the same view, you will need to copy the Salutation (eApps) business component, rename it, add the copy to the business object associated with the view, and then create a new applet in the view based on the copy.

Process of Adding a Message to the Salutation Applet

The process involved in adding a message to the salutation applet is the following:

- Selecting the salutation applet
- Selecting the rule set associated with the salutation applet
- Adding a rule to the rule set
- Providing a name, sequence, conditional expression, and inclusion expression (message) for the rule

NOTE: For salutation applets (based on the eApps business component), if HTML tags are used in Personalization rules, the personalization engine passes them on to the browser. This allows users to define formatted messages.

Modifying the Siebel eService Salutation Applet

You can modify the eService salutation applet within the Personalization Administration views. The following example shows how to add a message to the Siebel eService salutation applet that will request the user to register his or her product.

To modify the eService salutation applet

1. Navigate to Site Map > Personalization Administration > Applets.
2. Select Salutation Applet (eService).
3. Click the name of the rule set associated with Salutation Applet (eService).

   The Rule Sets view appears with eService Salutation selected. The rules belonging to the eService Salutation rule set appear below it in the Rules list.
4. Scroll down the rule list to the Message Self-Service rule, and add a new rule to the list.

5 Fill in the following fields under the More Info tab below the Rules list.

Field	Value
Name	Message Register
Sequence	13. This causes the rule to be evaluated after the Message Self-Service rule.
Rule Type	Expressions
Active	Select the check box.
Conditional Expression	((GetProfileAttr("Me.Is Anonymous") IS NULL) OR (GetProfileAttr("Me.Is Anonymous") = 'FALSE')) AND (GetProfileAttr ("Full Name") IS NOT NULL) This shows the expression only to registered users.
Description	Register your products. This field is optional.
Include Expression	" <img src='images/bullet_green.gif' align='top' > <b> Please register your products." This free-form HTML text displays a bullet with the message in bold.

The rule set is saved and the new rule is made active. It is not necessary to reload Siebel Personalization.

To view the result, log in to eService as SADMIN.

5 Run-time Events

Tracking run-time events allows the Siebel application to respond in real time to user actions. Business managers can configure the actions in response to any event.

Run-time Events and Action Sets

When a run-time event occurs that is associated with one or more action sets, the personalization engine performs the specified actions. These actions modify content and user profile attributes, with the potential of triggering personalization rules. You do not need to reconfigure or recompile business objects because the interaction occurs between Siebel Personalization and the user interface.

For example, you could watch for customers to remove products from their Siebel eSales Shopping Cart (that is, invoke the EmptyCart method of the Shopping Service business service), and then recommend a substitute product for them to consider.

Definitions of Events

Events are defined by:

- **Object Type.** The type of object to which the event occurs. This can be an application, business component, or applet.
- **Object Name.** The name of the application, business component, or applet to which the event occurs.
- **Event.** The specific event that happens to the object. The set of available events is different for different object types:
  - Application. The available events are:
    - **Login.** A user logs in to an application using the Siebel Mobile Web Client.
    - **Logout.** A user logs out from an application using the Siebel Mobile Web Client.
    - **SetAttribute.** A profile attribute is set.
    - **ViewActivated.** A view is activated.
    - **ViewDeactivated.** A view is deactivated.
    - **WebLogin.** A user logs in to the Siebel Web Client.
    - **WebLogout.** A user logs out from the Siebel Web Client.
    - **WebSessionEnd.** A Web session ends.

- ❑ **WebSessionStart.** A Web session begins.
- ❑ **WebTimeout.** A Web session times out.

- Business component. All business component events are named after corresponding Siebel VB BusComp functions. The available events are:
  - ❑ **Associate.** Triggered after an association is created when a record is added to a business component.
  - ❑ **ChangeRecord.** Triggered after the current row changes in the business component.
  - ❑ **CopyRecord.** Triggered after a new row is copied in the business component.
  - ❑ **DeleteRecord.** Triggered after a row is deleted in the business component.
  - ❑ **InvokeMethod.** Triggered after a method is called.
  - ❑ **NewRecord.** Triggered after a new row is added to the business component.
  - ❑ **PreAssociate.** Triggered before an association is created when a record is added to a business component.
  - ❑ **PreCopyRecord.** Triggered before a new row is copied in the business component.
  - ❑ **PreDeleteRecord.** Triggered before a row is deleted in the business component.
  - ❑ **PreGetFieldValue.** Triggered when the value of a business component field is accessed.
  - ❑ **PreInvokeMethod.** Triggered before a method is called.
  - ❑ **PreNewRecord.** Triggered before a new row is added to the business component.
  - ❑ **PreQuery.** Triggered before a query is executed.
  - ❑ **PreSetFieldValue.** Triggered when a value is written to the business component from the user interface.

    - **PreWriteRecord.** Triggered before a row is written to the database.
    - **Query.** Triggered after a query is executed.
    - **SetFieldValue.** Triggered after a value is written to the business component from the user interface.
    - **WriteRecord.** Triggered after a row is written to the database.
    - **WriteRecordNew.** Triggered after a new row is written to the database. Fires along with WriteRecord.
    - **WriteRecordUpdated.** Triggered after a row is updated in the database. Fires along with WriteRecord.

  - Applet. Applet events are triggered just after a method is invoked or an applet or record is displayed. The available events are:

    - **DisplayApplet.** An applet is displayed.
    - **DisplayRecord.** A record is displayed.
    - **InvokeMethod.** A method is invoked.

  For more information on events, see *Siebel Tools Reference*.

- **Subevent.** Used when the object type is BusComp or Applet and the event is InvokeMethod. The subevent is the name of the method to be monitored.

  The method name is found in Siebel Tools, in the Invoke Method property of the control that calls the method.

- **Conditional Expression.** Adds additional control. If the conditional expression evaluates to TRUE, the event is monitored.

- **Action Set.** What actions are taken when the event occurs. Action sets are defined in the Action Sets view under Runtime Events Administration.

  Most actions set or modify user profile attributes. These set expressions store information that is later used to tailor content delivery.

- **Sequence.** The order in which the action set associated with this event in this record executes, relative to other action sets associated with this event.

About Creating Action Sets

This section describes the following:

- Action Sets Defined
- Action Types Supported
- Creating Action Sets on page 77
- Creating Actions on page 77

Action Sets Defined

Action sets are groups of actions that are triggered by events. They are created in the Action Sets view under Runtime Events Administration, using the following steps:

- Create the action set that is to be associated with an event.
- Create the actions that are grouped in the action set.

Action Types Supported

The following action types are supported for all events:

- **Attribute Set.** Simple expression that sets user profile attributes. These are: setting to a constant value, auto-increment, auto-decrement, increment by a constant, and decrement by a constant.
- **BusService.** Invokes a method of the specified business service. The input to the business service is a property set with the following properties:
    - **Context.** Business service context.
    - **ActionSet.** Action set name.
    - **Action.** Action that caused the business service to be called.
    - **EventId.** Row ID of the event that caused the action to occur.
    - **Event Name.** Event that caused the action to occur.

  - **Sub Event.** Business service method to be monitored.
  - **Event Type.** Type of event.
  - **Object Name.** Object to which the event occurred.
  - **Business Component Name.** Used if the object type is Applet or BusComp. If Applet, the business component name is the name of the business component on which the applet is based. If BusComp, the business component name is the same as the object name.
- **Invoke Method.** Invokes a method on the business component that caused the event to occur.

Creating Action Sets

You create action sets in the Action Sets view under Runtime Events Administration.

To create an action set

1. Navigate to Site Map > Runtime Events Administration > Action Sets.
2. Add a new record to the Action Sets list.
3. Fill in the necessary fields.

Field	Description
Name	**Required.**
Start Date	Date when the action set takes effect.
End Date	Date when the action set stops being effective.
Active	**Required.** Select the check box to make the action set active.
Description	**Optional.** Provide more detail about the action set.

Creating Actions

You define actions to be triggered when an event occurs. Actions can modify content or user profile attributes, which in turn can modify the content displayed to the user.

Actions are created in the Action Sets view under Runtime Events Administration.

NOTE: After modifying personalization actions, you must refresh the object manager for the actions to take effect immediately. See “Clearing and Reloading Siebel Personalization” on page 97.

To create actions for an action set

1. Navigate to Site Map > Runtime Events Administration > Action Sets.
2. Select the action set for which you are creating actions.

3. Add a new record to the Actions list below the Action Sets list.

   The Actions form appears under the More Info tab below the Actions list.

4. Fill in the necessary fields under the More Info tab.

Field	Comments
Name	**Required.** Name for conveniently referring to the action.
Sequence	**Required.** Number describing the order in which the action occurs. Execution begins with the action with the lowest sequence number (such as 1). Actions with the same sequence number are executed in random order. Actions occur in sequence until all actions are completed.
Active	**Required.** Check the box to indicate whether the action will be triggered or not. Inactive actions are ignored when the event occurs. This is a quick way to turn off an action without changing the start and end dates.
Conditional Expression	If the conditional expression evaluates to TRUE, the action is triggered when the event occurs.
Start Date	The date on which the action starts occurring.
End Date	The date on which the action stops occurring.

Field	Comments
Action Type	**Required.** Select the type of action from the drop-down menu: ■ Attribute Set. Sets a user profile attribute. The most common action type. For example, `“ProdName”` is set to a product in the user’s Shopping Cart. Attribute Set uses the Profile Attribute, Set Operator, Value, Set Minimum, and Set Maximum fields. ■ BusService. Invokes a business service method. Can also be used to trigger a workflow process or to get profile attributes. The input argument is a property set. BusService uses the Business Service Name, Business Service Method, and Business Service Context fields. ■ InvokeMethod. Invokes a method on the object (applet, business object, or application) that caused the event to occur. InvokeMethod uses the Method Name and Method Argument fields.
Description	**Optional.** More detailed information about the action.
Profile Attribute	The user profile attribute to set.
Set Operator	One of five choices: ■ Set: sets a user attribute to an explicit value. Example: Set DidLogin to “Yes”. ■ Increment: increments a user attribute by an integer (default value, if Set Expression is blank, is 1). Example: Increment PagesView by 1. ■ Decrement: decrements a user attribute by an integer (default value, if Set Expression is blank, is 1). Example: Decrement PagesView by 10. ■ Add List: adds a value to a comma-delimited list. Example: add the name of the current product to the list of products viewed. ■ Remove List: removes a value from a comma-delimited list.

Field	Comments
Value	The expression to use with the set operator. This can be any expression that evaluates to something the Set Operator can use, usually a business component field. The Personalization Business Rules Designer is used to pick the business component field. Set, Add List, and Remove List must evaluate to a single value. Increment and Decrement must evaluate to an integer.
Set Minimum	For attributes that are numbers this creates a minimum value under which the attribute cannot fall.
Set Maximum	For attributes that are numbers this creates a maximum value over which the attribute cannot rise.
Business Service Name	Name of the business service to invoke, if the conditional expression evaluates to TRUE and the type is BusService.
Business Service Method	Method to invoke on the business service.

Business Service Context	Parameter to pass to the business service method. While a business service can take many name/value pair parameters, you can only pass one parameter—called Context—in Siebel Personalization.
	An example of a business service context is:
	`Action Type: BusService`
	`Business Service Name: Workflow Process Manager`
	`Business Service Method: RunProcess`
	`Business Service Context: "ProcessName", "My Workflow".`

Method Name	Method to invoke on the business component, if the conditional expression evaluates to TRUE and the action type is Invoke Method.
Method Argument	Parameter to pass the business component. You can only pass one argument.

Associating Events With Action Sets

You can set up events to be monitored by the system and to trigger actions when these events occur. The actions then modify content or user profile attributes or invoke business service methods, which can cause personalization rules to fire.

To associate an event with an action set

1. Navigate to Site Map > Runtime Events Administration > Events.
2. Add a new record, then fill in the fields.

Field	Description
Name	**Optional.** Select an event alias from the drop-down list. Selecting a name automatically populates the Object Type, Object Name, Event, and Subevent fields. This is based on the event template created in the Event Aliases list. For more information, see "Creating Event Aliases" on page 83.
Sequence	**Required.** An event can trigger multiple action sets. Enter numbers in this field to control when the action set associated with this event in this record executes relative to other action sets associated with this event.
Object Type	**Required.** Select from the drop-down list: ■ Application ■ BusComp ■ Applet
Object Name	The name of the application, business component, or applet (depending on the object type) to which the event occurs.
Event	**Required.** Select from the drop-down list. The choices depend on which object type you choose.
Subevent	Fill in Subevent only when the object type is BusComp or Applet and the event is InvokeMethod. In this case, the subevent is the name of the method to be monitored.

Field	Description
Conditional Expression	Enter a conditional expression if necessary to add additional control to whether the event is monitored.
Action Set Name	**Required.** Select an action set to run when the event occurs.

Creating Event Aliases

An event alias is a template that you can use to create commonly occurring events. Event aliases are created in the Event Aliases list under Runtime Events Administration.

Event aliases consist of a meaningful name and the minimum set of objects that define an event: Object Type, Object Name, Event, and Subevent. When you pick an event alias in the Name field of the Events form, the fields for the objects that define the event are automatically completed.

To create an event alias

1. Navigate to Site Map > Runtime Events Administration > Event Aliases.
2. Add a new record, then fill in the necessary fields.

Field	Description
Name	Provide a useful name for the event alias.
Object Type	**Required.** Select from the drop-down list: ■ Application ■ BusComp ■ Applet
Object Name	The name of the object to which the event occurs.
Event	**Required.** Select from the drop-down list. The choices depend on which object type you choose.
Subevent	Fill in Subevent only when the object type is BusComp or Applet and the event is InvokeMethod. In this case, the subevent is the name of the method to be monitored.

Testing 6

This chapter explains the ways in which you test your personalization rules.

Testing Personalization Rules

Testing your personalization rules is important for evaluating their effectiveness and correctness. Siebel Personalization allows you to test your personalization rules in a development environment before they are deployed in a production environment, using the following features:

- **Test mode.** For evaluating personalization rules.

  See "About Test Mode" on page 87 and "Using Test Mode to Test Personalization Rules" on page 88.

- **Log file.** For recording personalization events.

  See "Enabling Personalization Event Logging" on page 92.

- **XML export.** For exporting personalization objects to an XML file. This can later be imported into a production environment.

  See "Exporting and Importing Personalization Data" on page 95.

- **Reload personalization.** For updating the object manager with changes made to personalization rules.

  See "Clearing and Reloading Siebel Personalization" on page 97.

About Test Mode

The Test view under Personalization Administration allows you to set up a test mode to test personalization rules. The test mode allows you to set up the profiles of test users, specify the test application name and view, and then launch a new instance of the application to test personalization rules.

Setting Up the Test Mode Process

The following steps are involved in testing the personalization setup:

- Specifying the login and password of the primary user. The primary user is the person logged into the application, for example, a Siebel Call Center agent or Siebel eSales registered user.

  You can also load a saved setup file by clicking Open.

- Specifying the login of the secondary user, if desired. An example of a secondary user is a customer speaking by telephone with the call center agent.
- Clicking Load to load the persistent user profile attributes of the primary user into the Primary User Attributes list and those of the secondary user into the Secondary User Attributes list.
- Editing the persistent user profile attributes for both users. All changes made to the persistent user profile attributes are written only to the test setup, not to the Siebel database.
- Entering dynamic user profile attributes for both users.
- Specifying the application to test by filling in the URL for launching the application.
- Specifying the view if you want to test the personalization of a specific view.
- Clicking Test to get instructions for opening a new instance of the application to test personalization rules.
- Testing the personalization rules.
- Clicking Save to save the test setup.

Using Test Mode to Test Personalization Rules

Personalization rules can be tested in a staging environment before being used in the production environment. This is done in the Test view under Personalization Administration.

To test personalization rules

1. Navigate to Site Map > Personalization Administration > Test.

2 Fill in the following fields, and then click Load.

You can also click Open to use a saved test setup.

Field	Description
Primary User Login	**Required.** User who is logged in to the application, such as a call center agent.
Primary User Password	**Required.**
Secondary User Login	**Optional.** Someone who interacts with the primary user, such as a customer speaking with the call center agent.

Test Application	Path to the test application.
	Use a URL for the Siebel Web Client, for example:
	`http://<machine_name>/callcenter`
	Use a command string for the Siebel Mobile Web Client, for example:
	`D:\<Siebel_install_dir>\BIN\siebel.exe /l language_code /c “D:\<Siebel_install_dir>\bin\ENU\config_file.cfg” /d data_source`
	where:
	■ *`<Siebel_install_dir>`* is the full path to the client installation directory
	■ *`language_code`* is the three-letter code for the language, for example, `ENU` for US English and `JPN` for Japanese
	■ *`config_file`* is the application configuration file, for example, `uagent` for Siebel Call Center and `siebel` for Siebel Sales
	■ *`data_source`* is the database to which to connect: Local, Sample, or ServerDataSrc

Test View	**Optional.** If not specified, the screen will be blank when the test application launches. You will have to navigate to a view.

The persistent user profile attributes of the primary user are loaded into the Primary User Attributes list with a `Me.` prefix. Those of the secondary user are loaded into the Secondary User Attributes list with a `You.` prefix.

Person-related attributes have the value `Person` in the Source field. Organization-related attributes have the value `Organization`.

3. Select an attribute to edit its value.

   The Name and Source fields are read-only for persistent user profile attributes.

4. Add a new record under Primary User Attributes or Secondary User Attributes to add a new dynamic user profile attribute.

5. Fill in the Name and Value fields. The Source field is read-only and has the value `Dynamic`.

6. Click Test under the Test tab and take the following action:

   - If you are using a connected Web Client (accessed though an appropriate URL), the Test Mode dialog box appears, which prompts users to open a new browser window (by using a desktop icon or the Start menu, if using Windows). The Test Mode dialog box provides a URL to paste into the new browser window.

   - If you are using a Mobile Web Client, a new instance of the application opens. Go to Step 9 on page 91.

7. Copy the URL shown in the Test Mode dialog box, and then click OK.

8 Open a new browser window, and then paste the URL into it.

A new instance of the specified application launches. If the test view was not specified, the screen will be blank. You will have to navigate to a view.

9 After testing personalization rules, click Save in the Test view to save the test setup.

A dialog box appears prompting you to specify a file name. The file is saved to the Siebel File System.

Using the Log File to Test Personalization Rules

A record of personalization events that occur can be written to a log file. This log can be useful for testing and diagnosing personalization rules.

Running personalization with the log turned on does have a performance impact. Also, log files can become extremely large, which has an impact on disk space. You should not enable the log in production environments for longer than it takes to acquire the information or do the testing you need.

Enabling Personalization Event Logging

1. If using the Siebel Mobile Web Client, add an entry to the [Siebel] section of the configuration file for your application as follows:

   `PersonalizationLog = <directory path for log file + file name>`

   For example,

   ```
   PersonalizationLog = C:\Temp\Personalization.txt
   ```

2. If using the Siebel Web Client, navigate to Server Administration > Components.
3. Select the appropriate object manager.
4. Click the Component Parameters tab.
5. Select the Application Personal Log parameter.
6. Edit the following fields.

Field	Value
Current Value	Directory path for the log file
Value on Restart	Directory path for the log file

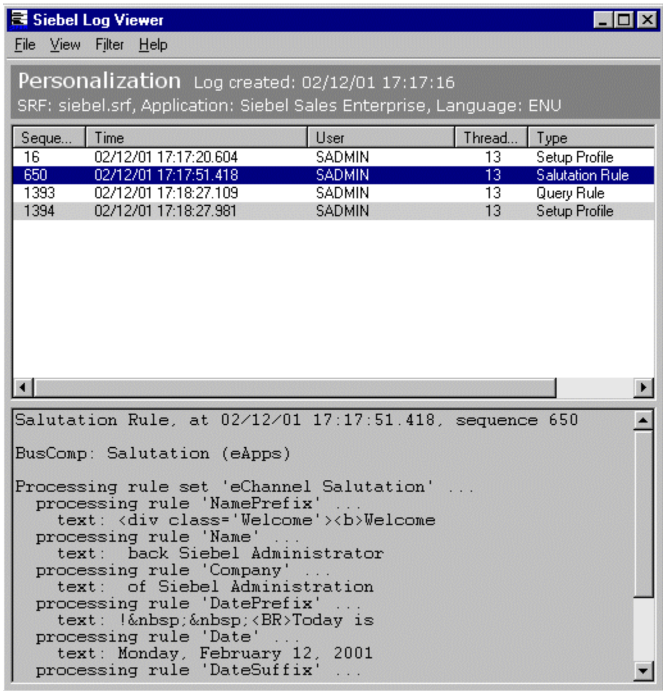
Testing Siebel Personalization Using the Log File

You can use the EnablePersonalization and EnableRuntimeEvents object manager parameters to test Siebel Personalization. These parameters are found in the `srvrdefs.dat` file. Navigate to Site Map > Server Administration > Components > Component Parameters to change them.

NOTE: These parameters can also be added to the application configuration file.

To test Siebel Personalization using the log file

1. Run the personalized application with the log enabled.
2. Open the log file using the graphical application SSFLOG.EXE, which is located in the <*Siebel_install_dir*>`\bin` directory, and shown in the following illustration.



Personalization Log created: 02/12/01 17:17:16
SRF: siebel.srf, Application: Siebel Sales Enterprise, Language: ENU

Seque...	Time	User	Thread...	Type
16	02/12/01 17:17:20.604	SADMIN	13	Setup Profile
650	02/12/01 17:17:51.418	SADMIN	13	Salutation Rule
1393	02/12/01 17:18:27.109	SADMIN	13	Query Rule
1394	02/12/01 17:18:27.981	SADMIN	13	Setup Profile

```
Salutation Rule, at 02/12/01 17:17:51.418, sequence 650

BusComp: Salutation (eApps)

Processing rule set 'eChannel Salutation' ...
  processing rule 'NamePrefix' ...
    text: <div class='Welcome'><b>Welcome
  processing rule 'Name' ...
    text:  back Siebel Administrator
  processing rule 'Company' ...
    text:  of Siebel Administration
  processing rule 'DatePrefix' ...
    text: !&nbsp;&nbsp;<BR>Today is
  processing rule 'Date' ...
    text: Monday, February 12, 2001
  processing rule 'DateSuffix' ...
```

3 Examine the log file under the following conditions.

Condition	What to Look For

EnablePersonalization = FALSE
EnableRuntimeEvents = FALSE
	■ Persistent user profile attributes are not loaded.
	■ Features based on personalization rules do not work.
	■ No run-time events occur.

EnablePersonalization = FALSE
EnableRuntimeEvents = TRUE
	■ Features based on personalization rules do not work.
	■ Persistent user profile attributes are not loaded.
	■ Run-time events occur.

EnablePersonalization = TRUE
EnableRuntimeEvents = FALSE
	■ Features based on run-time events do not work. Some personalization features also might not work, as they might depend on events.
	■ Persistent user profile attributes are loaded.
	■ Run-time events do not occur.

EnablePersonalization = TRUE
EnableRuntimeEvents = TRUE
	■ Siebel Personalization works properly.
	■ Persistent user profile attributes are loaded.
	■ Run-time events occur.

Exporting and Importing Personalization Data

Personalization rules, events, and actions can be exported as an XML file for later importation into another Siebel environment.

Exporting Personalization Data as an XML File

All personalization data rules, events, and actions are exported in one XML file.

To export personalization data as an XML file

1. Navigate to any Personalization Administration view, for example, Views.
2. Click the menu button, and then choose XML Export.
3. In the File Download window, select the Save this file to disk radio button, and then click OK.

   A dialog box appears prompting you to specify a file name and storage location. The default file name is `personalization.ooo.xml`.
4. Save the XML file.

Importing Personalization Data

Personalization rules, events, and actions can be imported from an XML file generated by a previous export.

NOTE: This process can take several minutes.

To import personalization data from an XML file

1. Navigate to any Personalization Administration view, for example, Views.
2. Click the menu button, and then choose XML Import.
3. Click Browse.

   A dialog box appears prompting you for a file name.
4. Choose a file, and then click OK.
5. Click Submit.

   The personalization data is imported. When the process is finished, a message displays how many records had conflicts and were inserted, updated, and skipped.

Clearing and Reloading Siebel Personalization

The Siebel object manager caches personalization data for improved performance. To see the effect of personalization rule changes, you must reload the personalization data into the object manager.

Reloading Siebel Personalization for the Current Object Manager

For the current object manager, reloading Siebel Personalization is performed in the Views, Applets, or Rule Sets view under Personalization Administration.

To clear and reload Siebel Personalization

1. Navigate to Site Map > Personalization Administration > Views.
2. Click the menu button, and then choose Reload Personalization.

   The personalization data is reloaded.

Reloading Siebel Personalization for Other Object Managers

To propagate changes in Siebel Personalization to other object managers, use the Enterprise Configuration view under Server Administration.

For more information on working with object managers, see the *Siebel Server Installation Guide* for the operating system you are using.

To clear and reload Siebel Personalization

1. Navigate to Site Map > Server Administration > Enterprise Configuration.
2. Click the Component Definitions view tab.
3. In the Component Definitions list at the top of the view, select the object manager for which to clear and reload personalization rules, for example, Call Center Object Manager.
4. Click the menu button under the upper Component Definitions list and choose Start Reconfiguration.

   The value of the Definition State field for that object manager changes to Reconfiguring.

5 Select the desired object manager again, click the menu button under the upper Component Definitions list, and then choose Commit Reconfiguration.

When the definition state changes to Active, the personalization data has been reloaded.

Siebel Query Language A

This chapter describes the operators that are available for building condition expressions and the functions that are specific to Siebel Personalization.

Operators

The following operators are available for building condition expressions. For more information on operators, conditions, and expressions, see *Siebel Tools Reference*.

Arithmetic Operators

Operator	Description
+	Add
–	Subtract
*	Multiply
/	Divide
^	Power

Comparison Operators

Operator	Description
=	Equal to a specific value.
< >	Not equal to a specific value.
>	Greater than a specific value.
<	Less than a specific value.
> =	Greater than or equal to a specific value.
< =	Less than or equal to a specific value.
~	NOT. Used to exclude values.
IS NULL	Finds blank fields. Using the = operator without any accompanying value does not give you the same results as IS NULL.
IS NOT NULL	Finds non-blank fields.

Logical Operators

Operator	Description
AND	All conditions connected by AND must be TRUE.
OR	At least one condition connected by OR must be TRUE.
NOT	The condition modified by NOT must be FALSE.

Pattern Matching Operators

Operator	Description
*	(asterisk) Wildcard for any number of characters before or after the value entered. Used in combination with LIKE.
?	(question mark) Wildcard for a single character. Used in combination with LIKE.
LIKE	Combine with a string and one or more wildcard characters to find all values with string. For example: [Product Line] LIKE "*enti?m*" returns records where the Product Line field contains the string "enti?m" such as "The Pentium I Line" and "Pentiums".
NOT LIKE	Combine with a string and one or more wildcard characters to find all values without string.
EXISTS	Finds a value from a multivalue group. EXISTS also finds values in any child of the MVG. For example: EXISTS ([Product Component] = "4 Meg Ram") returns records in which attribute Product Component (an MVG) referred to is a child record with the value "4 Meg Ram".

Functions

The following functions are specific to Siebel Personalization.

String Functions

Function	Description
Left (string, integer)	Returns the left-most *n* characters in the text string or field. For example, `Left (“Adams”, 2)` returns “Ad”.

Mid (string, start [, integer])	Returns *n* characters from the string starting from the start location.
	For example:
	`Mid (“Adams”, 2, 2)`
	returns “da”
	`Mid (“Adams”, 2)`
	returns “dams”.

Right (string, integer)	Returns the right-most *n* characters in the text string or field. For example: `Right (“Adams”, 2)` returns “ms”.
Len (string)	Returns the length of character string. For example: `Len (“foo”)` returns 3.

Function	Description

InStr (string1, string2 [, start] [, compare])	The position of the first character of string2 in string1.
	[,start] is used to specify the position in string1 to begin the search, where the first character in the string is 1. If start is negative, InStr searches backwards.
	[,compare] is used to specify if the comparison of strings is case sensitive. Enter 0 for a case-sensitive search, enter 1 for a case-insensitive search.
	For example:
	`InStr (“foo”, “oo”)`
	returns 2.
	`InStr (“bda”, “Adams”, 2)`
	returns 2.
	`InStr (“BDA”, “Adams”, 2, 1)`
	returns 2.
	If the string is not found, the function returns 0.

InList (string1, comma-separated list)	Returns if string1 occurs one or more times in a comma-separated list. For example: `InList(“foo”, “abc,boo,foo”)` returns TRUE.

Conditional Functions

Function	Description
IfNull (expr1, expr2)	Returns expr1 if expr1 is not NULL or returns expr2 if expr1 is NULL. IfNull is the return type of its first argument even if the first argument is NULL. For example: `IfNull (“”, “foo”)` returns “foo”.
IIf (testExpr, expr1, expr2)	If testExpr is TRUE, returns expr1’s value, otherwise returns expr2’s value. IfNull is the return type of its first argument even if the first argument is NULL. The second argument is converted to the type of the first argument before its value is returned. For example: `IIf ([Last Name] IS NULL, “foo”, “boo”)` returns foo if [Last Name] is NULL.
IsPrimary ()	For MVGs, returns if the current record is the primary record. IsPrimary () returns TRUE if the current address in an MVG address field is the primary address.
BCHasRow (BO, BC, search_expr, visibility)	Returns TRUE if the business component BC that is a part of business object BO has any rows after applying the search_expr and visibility. Evaluates to TRUE or FALSE depending on whether a given business component would return any records given the search specification and visibility. For example: `BCHasRow (“Contact”, “Contact”, “A*”, “Organization”)` returns TRUE if there are 1 or more rows of contacts. For example, you could hide an applet based on how many records it displayed. However, hiding and showing detail applets using BCHasRow is not possible.

Lookup Functions

Function	Description
Lookup (type, value)	Finds a row in the List of Values table (S_LST_OF_VAL) where the type column matches the type argument and the value column matches the value argument. The function returns the value of the ORDER_BY column for that row. For example: `Lookup (“MR_MS”, “Ms.”)` returns “2”.
LookupExpr (type, value_expr)	Searches the rows in the List of Values table (S_LST_OF_VAL) where the TYPE column matches the type argument. LookupExpr evaluates the contents of the VALUE column treated as an expression. Returns the value of the ORDER_BY column for the first row for which the expression evaluates to TRUE. For example: `LookupExpr (“MR_MS”, “M*”)` returns “1”.
LookupName (type, display_name)	Finds a row in the List of Values table (S_LST_OF_VAL) where the type column matches the type argument and the display value column matches the display_value argument. The function returns the language independent code. For example: `LookupName (“MR_MS”, “Ms.”)` returns “Ms.”
ParentFieldValue (field_name)	The value of the field_name field in the parent business component. The result does not change if the parent row is updated. The parent business component field must be exported by using Link Specification. For example: for the Opportunity business component `ParentFieldValue(“Account”)` returns “Name”.

Function	Description
FieldValue (field_name)	Return value of field in the business component as a string. For example: for Contacts business component for Henry Kim. For example: `FieldValue (“Last Name”)` returns “Kim”.
LookupValue (type, lang_ind_code)	Finds a row in the List of Values table (S_LST_OF_VAL) where the type column matches the type argument and the Language Independent Code column matches the language independent code argument. The function returns the display value. For example: `LookupValue (“MR_MS”, “Ms.”)` returns “Ms.”

Translation Functions

Function	Description
LookupTransVal (lang_ind_code, table, type)	For a multilingual list of values, given a language independent code, translation table (must be S_LST_OF_VAL), and LOV type, returns the display value. For example: `LookupTransVal (“Mr.”, “S_LST_OF_VAL”, “MR_MS”` returns “Mr.”
LookupTransCode (display_value, table, type)	For a multilingual list of values, given a display value, translation table (must be S_LST_OF_VAL), and LOV type, returns the language independent code. For example: `LookupTransCode (“Mr.”, “S_LST_OF_VAL”, “MR_MS”` returns “Mr.”

Function	Description
LookupTranslation ([field])	For the new multilingual data feature. Takes a field as an argument and if the field has a translation returns the translation. Otherwise returns the value of the field. For example: `LookupTranslation([Name])` returns “Sam” if the current language is English.
Preference (category, pref_name)	Finds the user preference for the category and preference name specified. `Preference (“User Interface”, “StyleSheet”)` returns the user-specified style sheet preference.
LoginName()	Returns the login ID of the logged in user.

Search Functions

Function	Description
FindOneOf (string1, string2)	Returns 1-based index of the first instance in string1 of a character in string2. For example: `FindOneOf (“abcdef”, “xyzc”)` returns 3.
FindNoneOf (string1, string2)	Returns 1-based index of the first instance in string1 which does not match any character in string2. For example: `FindNoneOf (“abcdef”, “xyzc”)` returns 1.

Math Functions

Function	Description
Sum (mvfield)	Sums the values from a field in child records. The child record being summed from must be defined as a multivalue field that is part of a multivalue group that is associated with the business component of the field being summed to. For example: `Sum ([Number of Employees])` gives the sum of all the employees at different locations for a company.
Count (mvfield)	Returns the number of rows in the multi-value group defined by the mvfield. For example: `Count([Number of Employees])` gives the number of employees at different locations for a company.
ToChar (number, format)	Returns a string that converts a number to a format specified by the format parameter. For details on the supported formats, see *Siebel Tools Reference*. For example: `ToChar (10, “##.##”)` returns “10.00”.
Min (mvfield)	Returns the minimum value from a field in child records. The child record being examined from must be defined as a multivalue field that is part of a multivalue group that is associated with the business component of the field being evaluated. For example: `Min ([Number of Employees])` gives the minimum number of employees of all the locations.

Function	Description
Max (mvfield)	Returns the Maximum value from a field in child records. The child record being examined from must be defined as a multivalue field that is part of a multivalue group that is associated with the business component of the field being evaluated. For example: `Max ([Number of Employees])` gives the maximum number of employees of all the locations.
InvokeServiceMethod (name, method, context, returnProperty)	Returns the value of the return property from the returnProperty set of the specified business service after invoking the method with the context. For example: `InvokeServiceMethod (“BusServ”, “PersonalizationMethod”, “Key1=a,Key2=2”, “ReturnProperty”)` invokes the business service method PersonalizationMethod in business service BusServ passes it the context Key1 = a,Key2 = 2 and returns the value set by the business service in the property ReturnProperty.
LN (num)	Returns the natural log of the num. For example: `LN (10)` returns 2.30.
GetNumBCRows (BO, BC, search_expr, visibility)	Retuns the number of rows business component BC that is a part of business object BO has after applying the search_expr and visibility. For example: `GetNumBCRows (“Contact”, “Contact”, “A*”, “Organization”)` returns the number of rows that match the criteria.

Date and Time Functions

Function	Description
JulianDay ()	Equal to the Oracle (and Sagent) Julian Day for all dates in the 20th and 21st centuries. Assigns an absolute numeric value to every date so that calculations can be performed on dates.
JulianMonth ()	Equal to the JulianYear() * 12 + currentMonth, where January = 1.
JulianQtr ()	Equal to the JulianYear () * 4 + currentQuarter, where currentQuarter = [(currentMonth -1) / 3 + 1] rounded down to the next integer.
JulianWeek ()	JulianDay () / 7, rounded down to the next integer.
JulianYear ()	Equal to the current year + 4713.
Timestamp ()	Returns today's date and time. For example: 2/15/2001 11:15:22
Today ()	Retuns the current date. For example: 2/15/2001
UtcConvert (utc_date_time, time_zone	This function converts UTC datetime field value to local time. For example: `UtcConvert("12/14/2000 5:07:05 PM", "Eastern Time")` returns "12/14/2000 05:07:05 PM"

Profile Functions

Function	Description

GetProfileAttr(‘profile’)	Returns the profile attribute of the current user.
	For example:
	`GetProfileAttr ('Personal Country')`
	returns the country of the user.
	GetProfileAttr only works for fields explicitly defined in the Personalization Profile business component. GetProfileAttr does not work with system fields, which are not explicitly defined in the business component; it returns a NULL value for them.
	An exception is the Id system field. This particular system field is available to GetProfileAttr even though it is not in the Personalization Profile business component.
	If this function is called for an MVG, it returns just the primary value of the MVG. For example, if the MVG State has the values CA, MA, and GA, where CA is primary:
	`GetProfileAttr ('State')`
	returns CA.

GetProfileAttrAsInt(‘profile’)	Returns the profile attribute of the current user as an integer. For example: `GetProfileAttrAsInt ('Age')` returns the age of the user as an integer.

Function	Description

GetProfileAttrAsList (‘profile’)	Returns the MVG value as a list. It can be used in the EXISTS operator to create expressions that match MVG profile attributes with appropriate Siebel content.
	Outside the Exists operator. Returns a comma-separated list of the MVG values. For example, if the MVG State has the values CA, MA, and GA, where CA is primary:
	`GetProfileAttrAsList ('State')`
	returns CA, MA, GA.
	For a single value field. `GetProfileAttrAsList` returns the profile attribute value.
	Within the EXISTS operator. Returns the value profile attributes in the form expected by the user. For example, the EXISTS operator could be used this way:
	`EXISTS ([Targeted States] = GetProfileAttrAsList (“State”)]`
	This action would match the MVG business component field Targeted State against the MVG profile attribute State.

GetProfileAttrAsNum (‘profile’)	Returns the profile attribute of the current user as a number. For example: `GetProfileAttrAsNum ('Age')` returns the age of the user as a number.
GetProfileAttrAsDate (‘profile’)	Returns the profile attribute of the current user as a date. For example: `GetProfileAttr ('Birth Day')` returns the birthday of the user as a date.
GetProfileAttrAsTime (‘profile’)	Returns the profile attribute of the current user as time. For example: `GetProfileAttr ('Last Login Time')` returns the last login time of the user as time.

Function	Description
GetProfileAttrAsDateTime (profile_attrib)	Returns the profile attribute of the current user as date and time. For example: `GetProfileAttr ('Last Login Date')` returns the last login date and time of the user as date and time.
GetProfileAttrAsUtcDateTime (profile_attrib)	Returns the profile_attrib formatted in the UtcDateTime type. For example: `GetProfileAttrAsUtcDateTime ('Date')` returns date time in UTC format.

Attribute Functions

Function	Description
XAIsClass (classname)	Returns TRUE if the current object belongs to the class “classname” including subclasses. For example: `XAIsClass (“”Car””)` returns TRUE for Honda Accord.
GetXAVal (attributename)	Returns the value for the specified string-type attribute for the current object if any. For example: `GetXAVal (“”Seat Type””)` returns leather for a luxury car with leather seats.
GetXAValAsInt (attributename)	Returns the value for the specified integer-type attribute for the current object if any. For example: `GetXAValAsInt (“”Doors””)` returns 4 for a four-door sedan.

Function	Description
GetXAValAsNum (attributename)	Returns the value for the specified number-type attribute for the current object if any. For example: `GetXAValAsNum (“”Doors””)` returns 4 for a four-door sedan.
GetXAValAsDate (attributename)	Returns the value for the specified date-type attribute for the current object if any. For example: `GetXAValAsDate (“”Release Date””)` returns the release date attribute for the row as date.

Other Functions

Function	Description
SystemPreference (“Pref”)	Get value of system preference. These values are found in System Preferences in the Application Administration screen. For example: `SystemPreference (“”Auto Mgr Calendar Access””)` returns TRUE.
LoginId ()	Returns the Login ID of the logged in user.
BCName ()	Returns the name of the active business component. For example: `BCName ()` returns ““Account””.
Currency ()	Returns the currency for the position of the logged in user. For example: USD
LocalCurrency ()	Returns the currency for the machine. For example: JPY

Function	Description
Language ()	Language code (for example, ENU) that is the active client language setting set by the Language parameter in the .cfg file, or by the `/L` parameter when starting Siebel.
RowidToRowidNum ()	Converts the Row ID of a record to a number. For example: `RowidToRowidNum ()` returns the unique integer translation of Rowid.

Index