Table 25. SWE Commands

Supported Values	Short Format	Description	Required Args (with Description)	Optional Args (with Description)
GotoView	Gv	Goes to a Siebel view. If the **SWEPostnApplet** and **SWEPostnRowId** arguments are specified, then it executes a search for the specified row ID in the specified applet. **NOTE:** If the queried applet is part of a toggle cycle, then set SWEPostnApplet to the default (top) applet in the toggle cycle or the application displays an error, View: %1 does not contain applet: %2. For more information about applet toggles, see *Configuring Siebel Business Applications*. If **SWEQMApplet** and **SWEQMMethod** arguments are specified, then it invokes the method after going to the view.	**SWEView** - name of the view.	**SWEKeepContext** - if TRUE, keeps the current business object context, when requesting to a view based on the same business object. **SWEPostnApplet** - name of the applet on which to execute the search. **SWEPostnRowId** - row ID to search for. **SWEQMApplet** - name of the QueueMethod applet where the method (as specified in **SWEQMMethod**) is invoked after going to the view. **SWEQMMethod** - name of the QueueMethod method to be invoked. You can invoke only one method. **SWEQMArgs** - arguments of the QueueMethod method.

Table 25. SWE Commands

Supported Values	Short Format	Description	Required Args (with Description)	Optional Args (with Description)
InvokeMethod For a list of commonly used methods, see Table 27 on page 80.	Inv	Invokes a method on an applet, a business service, a business component, or the SWE application. Use the optional **SWEService**, **SWEBusComp**, and **SWEApplet** arguments to specify the Siebel object on which the method is invoked. If none of these are specified, then SWE invokes on the SWE application object, which currently supports a limited set of InvokeMethod such as Logoff, SortOrder, SaveQuery, and SaveQueryAs.	**SWEMethod** - name of the method.	**SWEService** - nameof the business service to invoke the method. **SWEBusComp** - name of the business component to invoke the method. **SWEApplet** - name of the applet to invoke the method. **SWEView** - name of the view to invoke the method
LoadService	None	Loads a business service on the server side.	**SWEService** - name of the business service to load.	None
Login	Lg	Loads the login view or login page. SWE first looks at the Acknowledgment Web View property of the application object in the repository for the login view to show. If not specified, then the default is the "Acknowledgment Web Page" property to show the login page.	None	None

Table 25. SWE Commands

Supported Values	Short Format	Description	Required Args (with Description)	Optional Args (with Description)
Logoff	Bye	Executes the database logoff, then shows the logoff view or page. SWE first looks at the Logoff Acknowledgment Web Page property of the application object in the repository for the login page to show. If none is specified, then SWE shows the login view or login page, depending on how you log in.	None	None
ReloadCT	None	Reloads personalization info. SWE loads the initial personalization on startup, and when the personalization rulesare changed, SWE does not update the info automatically since there is cost in performance, so SWE provides this command to reload the info.	None	None

SWE Commands Available in Siebel Open UI

You can use several SWE commands to display a Siebel portlet in Siebel Open UI. For security reasons, you can use only the GotoViewmethod to call a Siebel portlet from an external application. GotoPage and GotoPageTab are not applicable to Siebel Open UI. You can use the commands listed in Table 26 within a Siebel portlet. You cannot use them to call a portlet. For more information about these commands, see *Configuring Siebel Open UI*.

Table 26. SWE Commands Available in Siebel Open UI

Supported Values	Inside External Application	Called from UI Element Inside Portlet Container	Called from Outside Portlet Container
CanInvokeMethod	Yes	Yes	No
ExecuteLogin	Yes	Not applicable for this use case.	No
GotoView	Use only when invoked from the browser address bar by refresh or history navigation.	Yes	Yes
InvokeMethod	Yes	Yes	No
LoadService	Yes	Yes	No
Login	Yes	Not applicable to Siebel Open UI.	Not applicable (use SSO or similar)
Logoff	Yes	Not applicable to Siebel Open UI.	No
ReloadCT	Yes	Yes	No

SWE Methods

The InvokeMethod command allows you to invoke SWE methods on an applet, business component, business service, or application. Table 27 lists SWE methods commonly used with the InvokeMethod SWE command.

Table 27. SWE Methods

Supported Values	Description	Required Args (with Description)	Optional Args (with Description)
CollapseTreeItem	Used in a tree control to collapse an expanded item on the tree.	**SWETreeItem**: Specify the path of the item relative to root. The path is a string of the form *n*.*n*.*n*.*n*...where *n* is an index of an item within its level. The index starts from 1. Example: 1.1.2. **SWEView**: Name of the view. **SWEApplet**: Name of the applet.	None
CopyRecord	Performs initialization, then calls CopyRecord on the business component.	None	None
CreateRecord	Performs initialization, then calls NewRecord on the business component.	None	None
DeleteQuery	Deletes a named query.	**SweNamedQueries**: Specify the name of the named query to be deleted.	None
DeleteRecord	Deletes a record.	None	None
Drilldown	Drills down on the field as specified in the argument **SWEField**.	**SWEField**: Specify the name of the applet field that you want to drilldown on. The drilldown information is specified in the repository.	None

Table 27. SWE Methods

Supported Values	Description	Required Args (with Description)	Optional Args (with Description)
EditRecord	Changes the Applet Web Template from base mode to edit mode, so the record can be edited. Use EditRecord with applets running in standard interactivity. For applets running in high interactivity, it is not necessary tochange the Applet Web Template mode to edit the record. For high interactivity applets, use WriteRecord. Siebel Open UI uses high interactivity applets.	**SWESeq**: Specify the sequence number of the Edit template. You can have many Edit templates for an applet in Siebel Tools, each identified by the sequence number.	List of arguments with name and value, where the name specifies the field name and the value specifies the field query specification. Sets the field query specification before executing the query.
ExecuteQuery	Executes a query. The query specification of the fields is specified in the list of arguments.	None	None
ExecuteNamedQuery	Executes a predefined query (PDQ) on the current view. Use with standard interactivity applications.	**SWEQueryName** - name of the PDQ.	None
ExpandTreeItem	Used in a tree control to expand an item on the tree.	**SWETreeItem**: Specify the path of the item relative to root. The path is a string of the form *n*.*n*.*n*.*n*...where *n* is an index of an item within its level. The index starts from 1. Example: 1.1.2. **SWEView**: Name of the view. **SWEApplet**: Name of the applet.	None
GotoFirstSet	Goes to the first set of records. The number of rows in a set is specified in the repository.	None	None

Table 27. SWE Methods

Supported Values	Description	Required Args (with Description)	Optional Args (with Description)
GotoLastSet	Goes to the last set of records.	None	None
GotoNextSet	Goes to the next set of records.	None	None
GotoPreviousSet	Goes to the previous set of records.	None	None
GotoView	Goes to a Siebel view. If the **SWEPostnApplet** and **SWEPostnRowId** arguments are specified, then this method executes a search for the specified row ID in the specified applet. **NOTE:** If the queried applet is part of a toggle cycle, then set SWEPostnApplet to the default (top) applet in the toggle cycle or the application displays an error, View: %1 does not contain applet: %2. For more information about applet toggles, see *Configuring Siebel Business Applications*. If **SWEQMApplet** and **SWEQMMethod** arguments are specified, then this method invokes the method after going to the view.	**SWETargetView** - name of the view.	**SWEKeepContext** - if TRUE, then this method keeps the current business object when the user navigates to a view that uses the same business object. **SWEPostnApplet** - name of the applet on which to execute the search. **SWEPostnRowId** - row ID to search for. **SWEQMApplet** - name of the QueueMethod applet where the method (as specified in **SWEQMMethod**) is invoked after going to the view. **SWEQMMethod** - name of the QueueMethod method. The method to be invoked. You can invoke only one method. **SWEQMArgs** - arguments of the QueueMethod method.

Table 27. SWE Methods

Supported Values	Description	Required Args (with Description)	Optional Args (with Description)
Indent	For a hierarchical applet, moves the current record down the hierarchy by one level.	None	None
MoveDown	For a hierarchical applet, moves the current record down the hierarchy within the same level.	None	None
MoveUp	For a hierarchical applet, moves the current record up the hierarchy within the same level.	None	None
NewQuery	Begins a new query.	None	None
NewRecord	If the applet has an association applet, then this method shows the association popup applet. Otherwise, it creates a new record.	None	None
NextTreeItem	Used in a tree control to scroll the tree to the next set of record.	**SWETreeItem**: Specifies the path of the item relative to root. The path is a string of the form *n*.*n*.*n*.*n*...where *n* is an index of an item within its level.The index starts from 1. Example: 1.1.2. SWEView: Name of the view. **SWEApplet**: Name of the applet.	None
Outdent	For a hierarchical applet, moves the current record down the hierarchy by one level.	None	None
PickNone	Makes sure the parent applet field has nothing picked from the pick applet.	None	None

Table 27. SWE Methods

Supported Values	Description	Required Args (with Description)	Optional Args (with Description)
PickRecord	Picks the current row in a pick applet.	None	None
PositionOnRow	Positions the record as specified in the list of required arguments.	**SWEView**: Name of the view. **SWEApplet**: Name of the Applet. **SWERowId**: The row ID of the desired record. **SWESetRowCnt**: Sets the number of rows to be returned for XML requests. When used during PositionOnRow, the specified number of rows are returned, and the selected row remains highlighted. **SWEReqRowId**: Indicates that the row is required in the operation.	None
PostChanges	Sets the field values as specified in the list of arguments to therecord being created or edited.	None	List of arguments with name and value where the name specifies the field name and the value specifies the field value. Sets these field values before committing the record.
PreviousTreeItem	Used in a tree control to scroll the tree to the previous set of records.	**SWETreeItem**: Specify the path of the item relative to root. The path is a string of the form *n*.*n*.*n*.*n*...where *n* is an index of an item within its level. The index starts from 1. Example: 1.1.2. SWEView: Name of the view. **SWEApplet**: Name of the applet.	None

Table 27. SWE Methods

Supported Values	Description	Required Args (with Description)	Optional Args (with Description)
RefineQuery	Keeps the current field query specification and queries again.	None	None
SaveQueryAs	Saves the current query as a named query. The name is specified in the argument **_SweNamedQueries**.	**SweNamedQueries**: Specify the name tosave the query as.	None
SelectTreeItem	Used in a tree control to select an item of the tree.	**SWETreeItem**: Specifies the path of the item relative to root. The path is a string of the form *n*.*n*.*n*.*n*...where *n* is an index of an item within its level.The index starts from 1. Example: 1.1.2. SWEView: Name of the view. **SWEApplet**: Name of the applet.	None
SortAscending	Sorts the field as specified in the argument **SWEField** in ascending order.	**SWEField**: Specifies the name of the applet field that you want to sort in ascending order.	None
SortDescending	Sorts the field as specified in the argument **SWEField** in descending order.	**SWEField**: Specifies the name of the applet field that you want to sort in descending order.	None
ToggleTo	Toggles to a different toggle applet.	**SWESeq**: Sequence number of the toggle applet to toggle to.	None
UndoRecord	Undoes a record that is being created or edited.	None	None
WriteRecord	Commits a record that is being created or edited.	**SWERowId**: Is the row ID of the record to be saved. **SWEReqRowId**: Indicates that the row ID is required in the operation.	None

SWE Arguments

Table 28 lists some commonly used SWE arguments.

Table 28. SWE Arguments

URL Argument	Short Format	Description	Usage	Examples
SWEAC	None	Allows login manager to string two SWE commands in a single request. (Formerly known as SWEAuxCmd.)	SWECmd=ExecuteLogin (followed by) SWEAC=GotoPageTab	SWECmd=ExecuteLogin&SWEUserName=joe&SWEPassword=passwd&SWEAC=SWECmd=GotoPageTab&SWEScreen=Accounts+Screen&SWEReloadFrames=1
SWEBU	None	Indicates that a URL is a bookmarked URL. It is retrieved in the UI by using the Get Bookmark URL command.	SWEBU=1 (if used as a bookmark URL)	None
SWECount	C	Dynamically generates an index number for each hyperlink for the purpose of bookmarking each request.	SWEC=*n* (where *n* is a positive integernumber) (or) <ARG NAME="SWEC">*n*</ARG>	SWEC=1 (or) <ARG NAME="SWEC">1</ARG>
SWEDataOnly	None	Discards all UI content (including anchors) if set to TRUE.	SWEDataOnly={TRUE|FALSE} (or) <ARG NAME=”SWEDataOnly”>TRUE|FALSE</ARG>	SWEDataOnly=TRUE (or) <ARG NAME=”SWEDataOnly”>TRUE</ARG>
SWEExclude	None	Uses the comma-separated UI element names specified as the value of the parameter to exclude UI elements from appearing in the output document.	SWEExclude=”list of names” (names can be MENU, SCREENBAR, TOOLBAR, THREADBAR, PAGEITEM, VIEWBAR) (or) <ARG NAME=”SWEExclude”>list of names</ARG>	SWEExclude=”MENU,SCREENBAR” (or) <ARG NAME=”SWEExclude”>MENU,SCREENBAR</ARG>

Table 28. SWE Arguments

URL Argument	Short Format	Description	Usage	Examples
SWEField	F	Specifies the name of the applet field.	`SWEField=<field name>` (or) `<ARG NAME="SWEField">field name</ARG>`	`SWEField=Revenue` (or) `<ARG NAME="SWEField">Revenue</ARG>`
SWEFullRefresh	None	Forces a full refresh of the Siebel Web Client, for applications deployed in Siebel Open UI or high interactivity. Used by the Siebel Open UI or high interactivity client to send a SWE command to load the completely. Typically used for session interleaving from a non-Siebel session.	`SWEFullRefresh={TRUE|FALSE}` (or) `<ARG NAME="SWEFullRefresh">TRUE|FALSE</ARG>`	`SWEFullRefresh=TRUE` (or) `<ARG NAME="SWEFullRefresh">TRUE</ARG>`
SWEGetApplet	None	This parameter is used to filter the outbound XML document so only the applet named as the value of the parameter is allowed in the output. All other document content is discarded.	`SWEGetApplet=<name of the applet>` (or) `<ARG NAME="SWEGetApplet">name of the applet</ARG>`	`SWEGetApplet=Account+List+Applet` (or) `<ARG NAME="SWEGetApplet">Account List Applet</ARG>`
SWEGetPDQ	None	Discards all XML content and returns only PDQ list when set to TRUE.	`SWEGetPDQ={TRUE|FALSE}` (or) `<ARG NAME="SWEGetPDQ">TRUE|FALSE</ARG>`	`SWEGetPDQ=TRUE` (or) `<ARG NAME="SWEGetPDQ">TRUE</ARG>`

Table 28. SWE Arguments

URL Argument	Short Format	Description	Usage	Examples
SWEKeepContext	Kx	Keeps the current business object if going to a view that uses the same business object, if set to TRUE.	`SWEKeepContext={TRUE|FALSE}` (or) `<ARG NAME=”SWEKeepContext”>TRUE|FALSE</ARG>`	`SWEKeepContext=TRUE` (or) `<ARG NAME=”SWEKeepContext”>TRUE</ARG>`
SWENeedContext	Nct	Skips restoring the state of the view, applet, business object, and business component when going back to a previously viewed page, if set to FALSE. Default is TRUE for a view or applet and FALSE for a Web page.	`SWENeedContext={TRUE|FALSE}` (or) `<ARG NAME=”SWENeedContext”>TRUE|FALSE</ARG>`	`SWENeedContext=TRUE` (or) `<ARG NAME=”SWENeedContext”>TRUE</ARG>`
SWENoAnchor	None	Discards all anchors if set to TRUE.	`SWENOAnchor={TRUE|FALSE}` (or) `<ARG NAME=”SWENoAnchor”>TRUE|FALSE</ARG>`	`SWENoAnchor=TRUE` (or) `<ARG NAME=”SWENoAnchor”>TRUE</ARG>`
SWEReloadFrames	RF	Forces the reloading of all HTML frames when set to TRUE.	`SWERF={TRUE|FALSE}` (or) `<ARG NAME="SWERF">TRUE|FALSE</ARG>`	`SWERF=TRUE` (or) `<ARG NAME="SWERF">TRUE</ARG>`
SWEReqRowId	Rqr	Needs to position to the row specified in the argument SWERowId, if set to TRUE.	`SWEReqRowId={TRUE|FALSE}` (or) `<ARG NAME=”SWEReqRowId”>TRUE|FALSE</ARG>`	`SWEReqRowId=TRUE` (or) `<ARG NAME=”SWEReqRowId”>TRUE</ARG>`
SWERows	Rs	Specifies the number of rows to be used as an attribute of an HTML frameset.	`SWERs=`*n* (where *n* is a positive integernumber) (or) `<ARG NAME="SWERs">n</ARG>`	`SWERs=1` (or) `<ARG NAME="SWERs">1</ARG>`

Table 28. SWE Arguments

URL Argument	Short Format	Description	Usage	Examples
SWERowId	R	The row ID of the record to position to.	SWERowId=<rowid><ARG NAME="SWERowId"> rowid</ARG>	SWERowId=12-XI46FG<ARG NAME="SWERowId">12-XI46FG</ARG>
SWERowIds	Rs	A string specifying the row ID of the parent business components.	SWERowIds=<string of rowids><ARG NAME="SWERowId"> string of rowids</ ARG>	SWERowIds=SWERowId0%3d12-61W25L<ARG NAME="SWERowId">SWERowId=12-61W25L</ ARG>
SWESetMarkup	None	Temporarily sets the markup language to use in the output document.	SWESetMarkup=<name of the markup language><ARG NAME="SWESetMarkup">markup language</ARG>	SWESetMarkup=HTML<ARG NAME="SWESetMarkup">HTML</ARG>
SWESetNoTempl	None	Disables the use of templates during the generation of the outbound document.	SWESetNoTempl={TRUE | FALSE}<ARG NAME="SWESetNoTempl"> TRUE|FALSE</ARG>	SWESetNoTempl=TRUE<ARG NAME="SWESetNoTempl">TRUE</ARG>
SWESetRowCnt	None	Temporarily sets the workset size or row number of list applets in the view.	SWESetRowCnt=<number of list rows><ARG NAME="SWESetRowCnt">number of list rows</ ARG>	SWESetRowCnt=50<ARG NAME="SWESetRowCnt">number of list rows</ARG>
SWEXslStyleSheet	None	Specifies the name of the XSLT style sheet to use to perform the XSLT on the XML output document.	SWEXslStyleSheet=<stylesheet name> (The style sheet needs to be in the application's webtempl directory.) <ARG NAME="SWEXslStyleSheet">name of the XSLT stylesheet</ARG>	SWEXslStyleSheet=ui.xsl<ARG NAME="SWEXslStyleSheet">ui.xsl</ARG>

Document Type Definition

This topic lists Document Type Definitions (DTD) for the inbound and outbound documents used with the XML Web Interface.

NOTE: The Siebel Open UI client supports HTML markup only. For more information, see "Overview of the XML Web Interface" on page 41.

Inbound DTD

The following is the DTD for the inbound documents used with the XML Web Interface.

```
<!ELEMENT    EXEC                (CMD, INFO*) >
<!ATTLIST    EXEC

   ATTR     CDATA   #IMPLIED
   PATH     CDATA   #IMPLIED
   TARGET   CDATA    #IMPLIED

>
<!ELEMENT   CMD                  (ARG*) >
<!ATTLIST   CMD

   NAME     CDATA   #REQUIRED
   VALUE    CDATA   #REQUIRED

>
<!ELEMENT   ARG    (#PCDATA) >
<!ATTLIST   ARG

   NAME     CDATA   #REQUIRED

>
<!ELEMENT   INFO   (#PCDATA) >
<!ATTLIST   INFO

   NAME     CDATA   #REQUIRED

>
```

Outbound DTD

The following is the DTD for the outbound documents used with the XML Web Interface.

```
<!ELEMENT   APPLICATION          (ERROR*, (USER_AGENT?, NAVIGATION_ELEMENTS*,
                                (SCREEN | APPLET | FORM | PDQ_BAR)* ), ERROR*) >
<!ATTLIST  APPLICATION

      NAME  CDATA                 #REQUIRED

>
<!ELEMENT   USER_AGENT           EMPTY>
<!ATTLIST   USER_AGENT

   MARKUP   CDATA                #REQUIRED
   TYPE     CDATA                 #IMPLIED

>
<!ELEMENT  NAVIGATION_ELEMENTS  (MENU*,
                                TOOL_BAR*,
                                SCREEN_BAR*,
                                THREAD_BAR*,
                                VIEW_BAR*,
                                 PAGE_ITEM*) >
```

```
<!ELEMENT   MENU                (MENU_ITEM | ERROR)* >
<!ATTLIST   MENU

   NAME     CDATA                #REQUIRED

>
<!ELEMENT  MENU_ITEM            (#PCDATA | ANCHOR | MENU_ITEM | ERROR)* >
<!ATTLIST  MENU_ITEM

   NAME    CDATA                #IMPLIED
   ENABLED (TRUE | FALSE)       #IMPLIED
   TYPE     CDATA                #IMPLIED

>
<!ELEMENT   ANCHOR              ((CMD, INFO*) | ERROR*) >
<!ATTLIST   ANCHOR

   ATTR    CDATA                #IMPLIED
   PATH    CDATA                IMPLIED
   TARGET   CDATA                #IMPLIED

>
<!ELEMENT   CMD                 (ARG*) >
<!ATTLIST   CMD

   NAME    CDATA                #REQUIRED
   VALUE    CDATA                #REQUIRED

>
<!ELEMENT   ARG                 (#PCDATA) >
<!ATTLIST   ARG

   NAME     CDATA                 #REQUIRED

>
<!ELEMENT   INFO                 (#PCDATA) >
<!ATTLIST   INFO

   NAME     CDATA                 #REQUIRED

>
<!ELEMENT   TOOL_BAR             (TOOL_ITEM | ERROR)* >
<!ATTLIST   TOOL_BAR

   NAME     CDATA                 #REQUIRED
   PATH     CDATA                  #IMPLIED

>
<!ELEMENT   TOOL_ITEM             (#PCDATA | ANCHOR | ERROR)* >
<!ATTLIST   TOOL_ITEM

   NAME       CDATA                #REQUIRED
   TYPE       CDATA                #REQUIRED
   ATTR       CDATA                #IMPLIED
   MAX_LENGTH CDATA#IMPLIED
```

```
>
<!ELEMENT   SCREEN_BAR            (SCREEN_TAB | VIEW_BAR | ERROR)* >
<!ELEMENT   SCREEN_TAB            (#PCDATA | VIEW_BAR | ANCHOR | ERROR)* >
<!ATTLIST   SCREEN_TAB

   NAME    CDATA                  #REQUIRED
   ACTIVE  (TRUE | FALSE)         "FALSE"
   CAPTION  CDATA                  #IMPLIED

>
<!ELEMENT  THREAD_BAR             (THREAD | ERROR)* >
<!ELEMENT  THREAD                 (#PCDATA | ANCHOR | ERROR)* >
<!ATTLIST  THREAD

   TITLE   CDATA                   #REQUIRED

>
<!ELEMENT  VIEW_BAR               (VIEW_TAB | ERROR)* >
<!ATTLIST   VIEW_BAR

   MODE    CDATA                  #IMPLIED
   SCREEN  CDATA                  #IMPLIED
   TYPE    CDATA                   #IMPLIED

>
<!ELEMENT  VIEW_TAB               (#PCDATA | ANCHOR | ERROR)* >
<!ATTLIST   VIEW_TAB

   NAME     CDATA                 #REQUIRED
   SELECTED (TRUE | FALSE)        "FALSE"
   TITLE     CDATA                 #IMPLIED

>
<!ELEMENT   PAGE_ITEM             (#PCDATA | ANCHOR | ERROR)* >
<!ATTLIST    PAGE_ITEM

   NAME    CDATA                  #REQUIRED
   ATTR    CDATA                  #IMPLIED
   CAPTION CDATA                  #IMPLIED
   TYPE    CDATA                   #REQUIRED

><!ELEMENT  SCREEN                (VIEW | ERROR*) >
<!ATTLIST   SCREEN

   NAME    CDATA                  #REQUIRED
   ACTIVE  (TRUE | FALSE)         "FALSE"
   CAPTION  CDATA                  #IMPLIED

>
<!ELEMENT   VIEW       (SUB_VIEW_BAR | PDQ_BAR | APPLET | IMG | FORM | ERROR)* >
<!ATTLIST   VIEW

   NAME     CDATA                  #REQUIRED
   ACTIVE  (TRUE | FALSE)          "FALSE"
   CATEGORY CDATA                  #IMPLIED
   TITLE    CDATA                   #IMPLIED
```

```
>
<!ELEMENT   APPLET          (FORM | CONTROL | CALENDAR | TREE | (LIST | (RS_HEADER,
                            RS_DATA)) | SORT_FIELD | APPLET_TOGGLE | ERROR)*  >

<!ATTLIST   APPLET

    NAME           CDATA            #REQUIRED
    ACTIVE         CDATA            #IMPLIED
    CLASS          CDATA            #IMPLIED
    ID             CDATA            #IMPLIED
    MODE           CDATA            #IMPLIED
    NO_DELETE      (TRUE | FALSE)   "FALSE"
    NO_EXEC_QUERY  (TRUE | FALSE)   "FALSE"
    NO_INSERT      (TRUE | FALSE)   "FALSE"
    NO_MERGE       (TRUE | FALSE)   "FALSE"
    NO_UPDATE      (TRUE | FALSE)   "FALSE"
    ROW_COUNTER    CDATA            #IMPLIED
    TITLE           CDATA            #IMPLIED

>
<!ELEMENT   FORM         ((CONTROL | CALENDAR | TREE | (LIST | (RS_HEADER, RS_DATA))
                 | SORT_FIELD | APPLET_TOGGLE | PDQ_BAR | SUB_VIEW_BAR)* | ERROR*) >

<!ATTLIST   FORM

    NAME    CDATA                   #IMPLIED
    ACTION  CDATA                   #IMPLIED
    ATTR    CDATA                   #IMPLIED
    METHOD  CDATA                   #IMPLIED
    TARGET  CDATA                    #IMPLIED

>
<!ELEMENT   CONTROL                 (#PCDATA | IMG | ANCHOR | PICK_LIST | ERROR)* >
<!ATTLIST   CONTROL

    NAME                CDATA           #REQUIRED
    ATTR                CDATA           #IMPLIED
    CALCULATED          (TRUE | FALSE)  "FALSE"
    CAPTION             CDATA           #IMPLIED
    DATATYPE            CDATA           #IMPLIED
    ENABLED             (TRUE | FALSE)  "FALSE"
    FIELD               CDATA           #IMPLIED
    FORMAT              CDATA           #IMPLIED
    HIDDEN              (TRUE | FALSE)  "FALSE"
    HTML_TYPE           CDATA           #IMPLIED
    ID                  CDATA           #IMPLIED
    MAX_LENGTH          CDATA           #IMPLIED
    NUMBER_BASED        (TRUE | FALSE)  "FALSE"
    READ_ONLY           (TRUE | FALSE)  "FALSE"
    REQUIRED            (TRUE | FALSE)  "FALSE"
    REQUIRED_INDICATOR  CDATA            #IMPLIED
    SCALE               CDATA            #IMPLIED
    TEXT_ALIGN          CDATA            #IMPLIED
```

```
    TEXT_BASED          (TRUE | FALSE)  "FALSE"
    TYPE                CDATA            #IMPLIED
    VARIABLE             CDATA            #IMPLIED

>
<!ELEMENT  PICK_LIST              (OPTION | ERROR)* >
<!ATTLIST  PICK_LIST

    NAME    CDATA                 #IMPLIED
    ATTR    CDATA                 #IMPLIED
    VALUE   CDATA                  #IMPLIED

>
<!ELEMENT  OPTION                 (#PCDATA | ERROR)* >
<!ATTLIST   OPTION

    CAPTION  CDATA                #IMPLIED
    SELECTED (TRUE | FALSE)        "FALSE"

>
<!ELEMENT   LIST                    ((RS_HEADER, RS_DATA) | ALERT | ERROR*) >

<!ELEMENT   RS_HEADER               (METHOD | COLUMN | ERROR)* >

<!ELEMENT   RS_DATA                 (ROW | ERROR)* >

<!ELEMENT   METHOD                  (#PCDATA | ANCHOR)* >

<!ATTLIST   METHOD

    NAME    CDATA                 #REQUIRED
    CAPTION CDATA                 #IMPLIED
    FIELD   CDATA                  #IMPLIED

>
<!ELEMENT   COLUMN                (METHOD | ERROR)* >
<!ATTLIST   COLUMN

    NAME            CDATA          #REQUIRED
    CALCULATED      (TRUE | FALSE)  "FALSE"
    DISPLAY_NAME    CDATA          #IMPLIED
    DATATYPE        CDATA          #IMPLIED
    FIELD           CDATA          #IMPLIED
    FORMAT          CDATA          #IMPLIED
    HIDDEN          (TRUE | FALSE)  "FALSE"
    HTML_TYPE       CDATA          #IMPLIED
    ID              CDATA          #IMPLIED
    LIST_EDITABLE   CDATA          #IMPLIED
    NUMBER_BASED    (TRUE | FALSE)  "FALSE"
    READ_ONLY       (TRUE | FALSE)  "FALSE"
    REQUIRED        (TRUE | FALSE)  "FALSE"
    SCALE           CDATA          #IMPLIED
    TEXT_ALIGN      CDATA          #IMPLIED
    TEXT_BASED      (TRUE | FALSE)  "FALSE"
```

```
    TEXT_LENGTH     CDATA           #IMPLIED
    TOTAL_REQUIRED  (TRUE | FALSE)  "FALSE"
    TYPE             CDATA           #IMPLIED

>
<!ELEMENT   ROW                     (#PCDATA | FIELD | ERROR)* >
<!ATTLIST   ROW

    ROWID    CDATA                   #REQUIRED
    SELECTED (TRUE | FALSE)           "FALSE"

>
<!ELEMENT   FIELD                   (#PCDATA | PICK_LIST | ANCHOR | ERROR)* >
<!ATTLIST   FIELD

    NAME    CDATA                   #REQUIRED
    VARIABLE CDATA                   #IMPLIED

>
<!ELEMENT   TREE                    (ITEM | ERROR)* >
<!ATTLIST   TREE

    NAME    CDATA                    #REQUIRED

>
<!ELEMENT   ITEM                     (#PCDATA | ACTION | ITEM | ERROR)* >

<!ATTLIST   ITEM

    ATTR      CDATA                 #IMPLIED
    CAPTION   CDATA                 #IMPLIED
    PATH      CDATA                 #REQUIRED
    SELECTED  (TRUE | FALSE)        "FALSE"
    TYPE       CDATA                 #IMPLIED

>
<!ELEMENT   ACTION                  (#PCDATA | ANCHOR)* >
<!ATTLIST   ACTION

    ATTR    CDATA                   #IMPLIED
    TYPE    CDATA                    #REQUIRED

>
<!ELEMENT   CALENDAR                EMPTY>
<!ATTLIST   CALENDAR

    TITLE    CDATA                   #IMPLIED

>
<!ELEMENT   SORT_FIELD              (PICK_LIST | ERROR)* >
<!ATTLIST   SORT_FIELD

    NAME     CDATA                  #REQUIRED
    SEQUENCE CDATA                   #IMPLIED
```

```
>
<!ELEMENT   APPLET_TOGGLE          (TOGGLE_ITEM | ERROR)* >
<!ATTLIST   APPLET_TOGGLE

   TYPE     CDATA                   #IMPLIED

>
<!ELEMENT   TOGGLE_ITEM            (#PCDATA | ANCHOR | ERROR)* >
<!ATTLIST   TOGGLE_ITEM

   APPLET_NAME  CDATA              #REQUIRED
   TITLE        CDATA              #IMPLIED
   SELECTED     (TRUE | FALSE)      "FALSE"

>
<!ELEMENT   SUB_VIEW_BAR            (VIEW_TAB | ERROR)* >

<!ELEMENT   PDQ_BAR                 (PDQ | ERROR)* >

<!ELEMENT   PDQ                    (#PCDATA | ANCHOR | ERROR)* >
<!ATTLIST   PDQ

   NAME     CDATA                  #REQUIRED
   SELECTED (TRUE | FALSE)          "FALSE"

>
<!ELEMENT   IMG                    (#PCDATA) >
<!ATTLIST   IMG

   ALT     CDATA                   #IMPLIED
   SRC     CDATA                    #IMPLIED

>
<!ELEMENT   ERROR                   (#PCDATA | ERROR)*  >

<!ELEMENT   ALERT                   (#PCDATA) >
```

Manipulating Siebel XML with XSL Style Sheets and XSLT

SWE can perform embedded XSL transformation on outbound XML documents. In this way, you can generate outbound documents in the desired markup language or format directly from SWE, without requiring a middle-tier server to perform the transformation. To do so, application developers must provide the XSL style sheets used for the transformation and specify the names of the style sheets to SWE.

NOTE: The Siebel Open UI client supports HTML markup only. For more information, see "Overview of the XML Web Interface" on page 41.

This topic contains the following information:

Defining SWE Style Sheet Tags

There are two ways in which you can request SWE to transform the outbound XML document into the desired format using XSLT. You can either pass in a query parameter SWEXslStyleSheet=*name-of-the-stylesheet*, or you can specify the style sheets to use in the Siebel templates by means of the `<swe:xsl-stylesheet>` tag. For more information, see "XML-Specific Template Tag" on page 97.

XML-Specific Template Tag

The XML-specific template tag looks like this:

```
<swe:xsl-stylesheet>
```

Purpose

Specifies the name of the XSLT style sheet to perform the XSLT on the XML output document. The style sheet must reside in the application's `webtempl` directory. There is only one `<swe:xsl-stylesheet>` tag for each view. If more than one `<swe:xslstylesheet>` tag is specified in the view, then the last tag found gets used.

Attributes

Two attributes are used with the XSLT style sheet:

- **name.** Specifies the name of the style sheet.
- **mode.** You can set the mode to either *process* or *embed*. When set to process, SWE performs XSLT processing on the XML output and sends the transformed document as the response back to the client. When this attribute is set to *embed*, SWE inserts an XML processing instruction in the beginning of the XML document for external XSLT processing.

Example

The following example illustrates how to specify the attributes for a style sheet.

```
<swe:xsl-stylesheet name= "table.xsl" mode= "process"/>
```

Sample XSL Style Sheet

The following XSL style sheet code sample is used to transform the WML-based Siebel Wireless application into HTML through the XML Web Interface. This code shows how a list view in the Wireless application is converted to HTML.

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

<xsl:output method="html" media-type="text/html"/>

<!-- This style sheet process the XML output for both the Splash screens and standard
views-->

<!-- ====================== Root Document Processing ========================-->

<!-- Document Root-->

<xsl:template match="/">

   <xsl:apply-templates select="//APPLICATION/SCREEN/VIEW/APPLET"></xsl:apply-
   templates>

</xsl:template>

<!-- ============================ View Processing ============================-->

<!-- List Base mode Template-->

<xsl:template match="APPLET">

<HTML>

   <BODY>

      <b>

      <!-- Applet Title Label-->

      <xsl:value-of select="CONTROL[@ID='1']"/>

      <!-- for calendar title -->

      <xsl:value-of select="CALENDAR/@TITLE"/>

      </b>

      <br></br>

      <!-- XML No Record found and other alerts -->

      <xsl:if test="string-length(ALERT)>0 and @CLASS='CSSFrameCalRerouteBase'">

      <xsl:value-of select="ALERT"/>

      <br></br>

      </xsl:if>

      <!-- Search and Title with data or other links -->

      <xsl:apply-templates select="CONTROL[@ID=2 or @ID=3 or @ID=4 or @ID=5 or @ID=6
      or @ID=7 or @ID=8 or @ID=9]"/>
```

```
        <!-- Separator line -->
        <xsl:apply-templates select="CONTROL[@ID=1000]"/>
        <!-- Display fields for list of records here-->
        <xsl:apply-templates select="LIST"></xsl:apply-templates>
        <xsl:if test="string-length(@ROW_COUNTER)>0">
        <xsl:value-of select="@ROW_COUNTER"></xsl:value-of>
        <br></br>
        </xsl:if>
        <!--  control link for New, Main Menu, etc.. -->
        <xsl:apply-templates select="CONTROL[@ID>=40 and @HTML_TYPE='Link']"/>
    </BODY>
</HTML>
</xsl:template>
<!-- ================= Control and Link Processing ====================-->
<xsl:template match="CONTROL">
    <xsl:choose>
        <xsl:when test="@HTML_TYPE='Link'">
            <xsl:call-template name="build_simple_link"></xsl:call-template>
        </xsl:when>
        <xsl:otherwise>
            <xsl:value-of select="."></xsl:value-of><br></br>
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>
<xsl:template name="build_simple_link">
    <xsl:variable name="link">
        <xsl:apply-templates select="ANCHOR"></xsl:apply-templates>
    </xsl:variable>

    <xsl:element name="A">
```

```
        <xsl:attribute name="HREF"><xsl:value-of select="$link"/></xsl:attribute>
        <xsl:value-of select="@CAPTION"/>
    </xsl:element>
    <br/>
</xsl:template>
<!-- ============================ List processing ==========================-->
<!-- LIST Template  builds a list of records -->
<xsl:template match="LIST">
    <!-- first get the URL from the RS_HEADER element-->
    <xsl:variable name="link">
        <xsl:apply-templates select="RS_HEADER/METHOD[@NAME='Drilldown']"/>
        </xsl:variable>
    <!-- capture the URL before the SWERowId parameter-->
    <xsl:variable name="link-prefix">
        <xsl:value-of select="substring-before($link,'R=')"/>
    </xsl:variable>
    <!-- capture the URL after the SWERowId parameter-->
    <xsl:variable name="link-suffix">
        <xsl:value-of select="substring-after($link,'R=')"/>
    </xsl:variable>
    <!-- capture the field with the drilldown enabled - use later to build drilldown
    -->
    <xsl:variable name="drilldowncontrol">
        <xsl:value-of select="RS_HEADER/METHOD[@NAME='Drilldown']/@FIELD"></
        xsl:value-of>
</xsl:variable>
    <!-- loop through the rows in the RS_DATA element -->
    <xsl:for-each select="RS_DATA/ROW">
        <!-- pickup the Row Id for the Row so we can rebuild the SWERowId URL parameter-
        ->
        <xsl:variable name="rowid">
```

```
            <xsl:value-of select="@ROWID"/>
         </xsl:variable>
         <!-- loop through each field and control in the Row -->
         <xsl:for-each select="FIELD|CONTROL">
            <xsl:choose>
               <!-- if the field is the drilldown field then create a link on the
         display data-->
               <xsl:when test="@NAME = $drilldowncontrol">
                  <xsl:element name="A">
                     <xsl:attribute name="HREF">
                        <xsl:value-of select="concat(normalize-space($link
         prefix),'R=',$rowid,$link-suffix)"/>&amp;F=<xsl:value-of select="@VARIABLE"/
         >
                     </xsl:attribute>
                     <xsl:value-of select="."></xsl:value-of>
                  </xsl:element>
               </xsl:when>
               <!-- otherwise just display the data as is-->
               <xsl:otherwise>
                  <xsl:value-of select="."></xsl:value-of>
               </xsl:otherwise>
            </xsl:choose>
            <!-- need a break if field is not empty -->
            <xsl:variable name="empty_field">
               <xsl:value-of select="."/>
            </xsl:variable>
            <xsl:if test="string-length($empty_field)!=0"><br></br></xsl:if>
         </xsl:for-each>
      </xsl:for-each>
   <!-- Show separator line only if has one or more record -->
      <xsl:variable name="row_data">
```

```
        <xsl:value-of select="normalize-space(RS_DATA/ROW)"/>
    </xsl:variable>
    <xsl:if test="string-length($row_data)>0">
        <xsl:text>- - - -</xsl:text><br></br>
    </xsl:if>
<!-- show More link only if there is next record set -->
    <xsl:variable name="more_link">
        <xsl:value-of select="normalize-space(RS_HEADER/METHOD[@NAME='GotoNextSet']/
        @CAPTION)"/>
    </xsl:variable>
    <xsl:if test="string-length($more_link)>0">
        <xsl:element name="A">
            <xsl:attribute name="HREF">
                <xsl:apply-templates select="RS_HEADER/METHOD[@NAME='GotoNextSet']">
            </xsl:attribute>
            <xsl:value-of select="$more_link"></xsl:value-of>
        </xsl:element>
        <br></br>
    </xsl:if>
</xsl:template>
<!-- ================== Anchor URL Processing ======================-->
<!-- ANCHOR Template builds the URL for drilldowns and links -->
<xsl:template match="ANCHOR">
    <xsl:text>start.swe?</xsl:text>
    <xsl:apply-templates select="CMD|INFO"/>
</xsl:template>
<xsl:template match="CMD">
    <xsl:value-of select="@NAME"/>=<xsl:value-of select="@VALUE"/>
    <xsl:apply-templates select="ARG"/>
</xsl:template>
```

```
<xsl:template match="ARG">
   <xsl:variable name="arg">
      <xsl:if test="string-length(normalize-space(.)) >0">
         <xsl:variable name="argstring">
            <xsl:choose>
            <xsl:when test="@NAME='Pu' or @NE='R' or @NAME='Rs'">
               <xsl:value-of select="translate(normalize-space(),'%2B','+')'"/>
            </xsl:when>
               <xsl:otherwise>
                  <xsl:value-of select="normalize-space()"/>
               </xsl:otherwise>
            </xsl:choose>
         </xsl:variable>
         <xsl:value-of select="$argstring"/>
      </xsl:if>
   </xsl:variable>
   <xsl:text>&amp;</xsl:text>
   <xsl:value-of select="@NAME"></xsl:value-of>=<xsl:value-of select="$arg"></
   xsl:value-of>
   <!--<xsl:text>&#38;</xsl:text>-->
   <!--<xsl:value-of select="@NAME"/>=<xsl:value-of
   select="translate($arg,'%2B','+')'"/>-->
</xsl:template>
<xsl:template match="INFO">
   <xsl:variable name="info">
      <xsl:if test="string-length(normalize-space(.)) >0">
         <!--<xsl:value-of select="."/>-->
         <xsl:value-of select="normalize-space(.)"/>
      </xsl:if>
   </xsl:variable>
   <xsl:text>&amp;</xsl:text>
```

```
      <xsl:value-of select="@NAME"/>=<xsl:value-of select="$info"/>

   </xsl:template>

   </xsl:stylesheet>
```

Sample XSLT

The following example shows how XSLT code snippets transform an XML response from SWE into HTML. The XSLT snippets are based on the XML response generated from the Query String example described in "Connecting to the XML Web Interface" on page 44.

```
<xsl:template match="/">
   <TABLE bgcolor="#CCCCFF" width="100%" cellpadding="2"
   cellspacing="0" Border="0" >
   <TBODY>
      <xsl:apply-templates select="//APPLET/LIST"/>
   </TBODY>
   </TABLE>
</xsl:template>

<xsl:template match="LIST">
   <xsl:apply-templates select="RS_HEADER"/>
   <xsl:apply-templates select="RS_DATA"/>
</xsl:template>

<xsl:template match="RS_HEADER">
   <TR>
      <xsl:for-each select="COLUMN">
      <xsl:if test="@NAME='Name'">
         <TD colspan="3" bgcolor="#CCCCFF" class="sub2viewon" width="60%">
         <B><xsl:value-of select="@DISPLAY_NAME"/></B
         </TD>
      </xsl:if>
      <xsl:if test="@NAME='Location'">
         <TD bgcolor="#CCCCFF" class="sub2viewon" width="40%">
         <B><xsl:value-of select="@DISPLAY_NAME"/></B>
         </TD>
      </xsl:if>
      </xsl:for-each>
   </TR>
</xsl:template>

<xsl:template match="RS_DATA">
   <xsl:for-each select="ROW">
      <TR>
         <xsl:for-each select="FIELD">
         <xsl:if test="@NAME='Name'">
            <TD bgcolor="#FFFFFF">
               <xsl:element name="IMG">
                  <xsl:attribute name="SRC">
                     portal_files/w.gif
                  </xsl:attribute>
```

```
            <xsl:attribute name="height">
               1
            </xsl:attribute>
            <xsl:attribute name="width">
               3
            </xsl:attribute>
         </xsl:element>
      </TD>
      <TD bgcolor="#FFFFFF" valign="top">
         <xsl:element name="IMG>
            <xsl:attribute name="SRC">
               portal_files/dot.gif
            </xsl:attribute>
            <xsl:attribute name="height">
               6
            </xsl:attribute>
            <xsl:attribute name="width">
               6
            </xsl:attribute>
         </xsl:element>
      </TD>
      <TD bgcolor="#FFFFFF" align="left" valign="top"
      width="60%">
         <xsl:choose>
            <xsl:when test="string-length(normalize
            space(.))> 0"
               <xsl:choose>
                  <xsl:when test="@NAME='Name'">
                  <xsl:call-template name="link"/>
                  </xsl:when>
                  <xsl:otherwise>
                  <xsl:value-of select="."/>
                  </xsl:otherwise>
               </xsl:choose>
            </xsl:when>
            <xsl:otherwise>
            <xsl:text> </xsl:text>
            </xsl:otherwise>
         </xsl:choose>
      </TD>
   </xsl:if>
   <xsl:if test="@NAME='Location'">
      <TD bgcolor="#FFFFFF" align="left" valign="top"
      width="40%">
         <xsl:choose>
            <xsl:when test="string-length(normalize-space(.))
            < 0">
               <xsl:choose>
                  <xsl:when test="@NAME='Name'">
                  <xsl:call-template name="link"/>
                  </xsl:when>
                  <xsl:otherwise>
                  <xsl:value-of select="."/>
                  </xsl:otherwise>
```

```
                  </xsl:choose>
                </xsl:when>
                <xsl:otherwise>
                <xsl:text> </xsl:text>
                </xsl:otherwise>
              </xsl:choose>
            </TD>
          </xsl:if>
          </xsl:for-each>
        </TR>
        <tr>
          <td colspan="4" width="40%"></td>
        </tr>
    </xsl:for-each>
</xsl:template>
```

5 Web Engine HTTP TXN Business Service

This chapter describes the Web Engine HTTP TXN Business Service. It contains the following information:

About the Web Engine HTTP TXN Business Service

HTTP provides several means to allow Web servers to obtain information from the browser. The most familiar example is when a user enters data into a form on a Web page and the data is sent to the Web server, which can access the value of each form field. This example illustrates sending form field parameters to the Web server with a `POST` method. In general, a browser can send cookies, headers, query string parameters, and form field parameters to the Web server. Web servers can also respond to the browser with cookies and custom headers. The Web Engine HTTP TXN Business Service allows Siebel Business Applications to retrieve or set cookies, headers, and query string and form field parameters.

The Web Engine HTTP TXN Business Service can be invoked by scripts or by workflow. The inbound HTTP request to the Siebel Web Engine (SWE) is parsed and the business service returns property sets containing cookies, headers, or parameters. In addition, server variables, which are not a part of the HTTP request header, can also be retrieved. The business service can also set a custom cookie or header in the HTTP response header generated by the SWE. The business service gives complete control over the request header received and the response header sent by the SWE.

For more information, see the following topics:

- "Web Engine HTTP TXN Business Service API" on page 108
- "Example of Using Web Engine HTTP TXN Business Service" on page 111
- "Sample Code for Invoking Methods of Web Engine HTTP TXN Business Service" on page 116

Web Engine HTTP TXN Business Service API

Table 29 lists the methods exposed by the Web Engine HTTP TXN Business Service.

Table 29. Web Engine HTTP TXN Business Service API

Method	Description	Parameters
GetAllRequestCookies	Retrieves all request cookies sent from the client to the server.	**InputArguments:** Ignored. **OutputArguments:** Property Set hierarchy. Each cookie is a child Property Set with the TYPE property set to the cookie name.
GetAllRequestHeaders	Retrieves all request headers sent from the client to the server.	**InputArguments:** Ignored. **OutputArguments:** Property Set containing the HTTP Parameter name-value pairs.
GetAllRequestParameters	Retrieves all request parameters sent from the client to the server.	**InputArguments:** Ignored. **OutputArguments:** Property Set containing the HTTP Parameter name-value pairs.
GetAllResponseCookies	Retrieves all response cookies sent from the server to the client.	**InputArguments:** Ignored. **OutputArguments:** Property Set hierarchy. Each cookie is a child Property Set with the TYPE property set to the cookie name.
GetAllResponseHeaders	Retrieves all response headers sent from the server to the client.	**InputArguments:** Ignored. **OutputArguments:** Property Set containing the HTTP Header name-value pairs.
GetAllServerVariables	Retrieves all server variables.	**InputArguments:** Ignored. **OutputArguments:** Property Set containing the Server Variable name-value pairs.
GetClientCertificate	Retrieves the client certificate info.	**InputArguments:** Ignored. **OutputArguments:** Property Set containing certificate name-value pairs. Currently only returns Common Name (CN) property of the certificate.

Table 29. Web Engine HTTP TXN Business Service API

Method	Description	Parameters
GetRequestCookies	Retrieves the request cookies named in InputArguments.	**InputArguments:** Property Set containing the cookie names to retrieve. **OutputArguments:** Property Set hierarchy. Each cookie is a child Property Set with the TYPE property set to the cookie name.
GetRequestHeaders	Retrieves the request headers named in InputArguments.	**InputArguments:** Property Set containing the header names to retrieve. **OutputArguments:** Property Set containing the HTTP Header name-value pairs.
GetRequestInfo	Retrieves the request Web Session, Headers, Cookies, Parameters and Client Certificate information in one call.	**InputArguments:** Ignored **OutputArguments:** Property Set hierarchy. Each section is a child Property Set with the TYPE property set to 'Headers', 'Cookies', 'Parameters' or 'ClientCertificate'. The Web Session information is simply stored as properties of OutputArguments.
GetRequestParameters	Retrieves the request parameters named in InputArguments.	**InputArguments:** Property Set containing the parameter names to retrieve. **OutputArguments:** Property Set containing the HTTP Parameter name-value pairs.
GetResponseCookies	Retrieves the response cookies named in InputArguments.	**InputArguments:** Property Set containing the cookie names to retrieve. **OutputArguments:** Property Set hierarchy. Each cookie is a child Property Set with the TYPE property set to the cookie name.
GetResponseHeaders	Retrieves the response headers named in InputArguments.	**InputArguments:** Property Set containing the header names to retrieve. **OutputArguments:** Property Set containing the HTTP Header name-value pairs.

Table 29. Web Engine HTTP TXN Business Service API

Method	Description	Parameters
GetResponseInfo	Retrieves the response Headers and Cookies in one call.	**InputArguments:** Ignored. **OutputArguments:** Property Set hierarchy. Each section is a child Property Set with the TYPE property set to 'Headers' or 'Cookies'. Content Type and Status aresimply stored as properties of OutputArguments.
GetServerVariables	Retrieves the server variables named in InputArguments.	**InputArguments:** Property Set containing the server variable names to retrieve. **OutputArguments:** Property Set containing the Server Variable name-value pairs.
GetWebSessionInfo	Retrieves the client's Web session information.	**InputArguments:** Ignored. **OutputArguments:** Property Set containing the Web session name-value pairs—SessionName; Cookie Name; SessionId; Web Session ID; SessionFrom (Value is 'URL' or 'COOKIE').
SetResponseCookies	Sets the response cookies to the values in InputArguments.	**InputArguments:** Property Set hierarchy. Each cookie is a child Property Set with the TYPE property set to the cookie name. The PERSISTENT property determines whether the cookiepersists between sessions. If the value is Y, then the cookie persists between browser sessions. Otherwise, the cookie exists for one session at a time. **OutputArguments:** Ignored.

Table 29. Web Engine HTTP TXN Business Service API

Method	Description	Parameters
SetResponseHeaders	Sets the response headers to the values in InputArguments.	**InputArguments:** Property Set containing the HTTP Header name-value pairs. **OutputArguments:** Ignored.
SetResponseInfo	Sets the response Headers and Cookies in one call.	**InputArguments:** Property Set hierarchy. Each section is a child Property Set with the TYPE property set to 'Headers' or 'Cookies'. Content Type and Status are simply stored as properties of InputArguments. **OutputArguments:** Ignored.

Example of Using Web Engine HTTP TXN Business Service

To invoke each method of the Web Engine HTTP TXN Business Service and write the results to a text file, use the following two procedures:

- "Adding Sample Code for Displaying Results of Using the Business Service" on page 111
- "Adding Sample Code for Invoking Methods of the Business Service" on page 113

Adding Sample Code for Displaying Results of Using the Business Service

The following procedure shows how to add sample code for displaying results of the Web Engine HTTP TXN Business Service.

To add sample code for displaying results of Web Engine HTTP TXN Business Service

1. In Oracle's Siebel Tools, navigate to the desired Applet object, in the Object Explorer.
2. Lock the project, if required.
3. Right click and select the Edit Server Script option.
4. Add the following three functions, individually to the declarations section:
   - WebApplet_OutputChildPropertySets
   - WebApplet_OutputProperties
   - WebApplet_OutputPropertySet

Sample Code Functions

Sample code for the WebApplet_OutputChildPropertySets Function:

```
function WebApplet_OutputChildPropertySets(oPropertySet, nLevel, fp)
{
var oChildPropSet;
var nChild = 0;

Clib.fputs('-------------------------------------\n',fp);
Clib.fputs('CHILD PROPERTY SETS\n',fp);
Clib.fputs('-------------------------------------\n',fp);

if ( oPropertySet.GetChildCount() == 0 )
{

Clib.fputs('(NONE)\n',fp);
}
else
{
for ( nChild = 0; ( nChild <= oPropertySet.GetChildCount() - 1 ) ; nChild++ )
{
oChildPropSet = oPropertySet.GetChild(nChild);
WebApplet_OutputPropertySet (oChildPropSet, nLevel+1, fp);
}
}
}
```

Sample code for the WebApplet_OutputProperties Function:

```
function WebApplet_OutputProperties(oPropertySet, nLevel , fp )
{
var strName;
var strValue;

Clib.fputs('-------------------------------------\n',fp);
Clib.fputs('PROPERTIES\n',fp);
Clib.fputs('-------------------------------------\n',fp);

if (oPropertySet.GetPropertyCount() == 0 )
{
Clib.fputs('(NONE)\n',fp);
}
else
{
strName = oPropertySet.GetFirstProperty();
while ( strName != '')
{
Clib.fputs(strName + ' : ' + oPropertySet.GetProperty(strName) + '\n' ,fp);
strName = oPropertySet.GetNextProperty();
}
}
}
```

Sample code for the WebApplet_OutputPropertySet Function:

```
function WebApplet_OutputPropertySet(oPropertySet, nLevel, fp )
{
Clib.fputs('\n',fp);
Clib.fputs('-------------------------------------\n',fp);
Clib.fputs('START' + ' ',fp);
Clib.fputs('LEVEL : ' + nLevel + '\n', fp);
Clib.fputs('-------------------------------------\n',fp);

Clib.fputs('TYPE  : ' + oPropertySet.GetType() + '\n',fp);
Clib.fputs('VALUE : ' + oPropertySet.GetValue() + '\n',fp);

WebApplet_OutputProperties(oPropertySet, nLevel, fp);
WebApplet_OutputChildPropertySets(oPropertySet, nLevel, fp);

Clib.fputs('-------------------------------------\n',fp);
Clib.fputs('END' + ' ',fp);
Clib.fputs('LEVEL : ' + nLevel + '\n',fp);
Clib.fputs('-------------------------------------\n',fp);
}
```

Adding Sample Code for Invoking Methods of the Business Service

The following procedure shows how to add sample code for invoking methods of the Web Engine HTTP TXN Business Service.

To add sample code for invoking methods of Web Engine HTTP TXN Business Service

1. Add the code from "Sample Code for Invoking Methods of Web Engine HTTP TXN Business Service" on page 116 to the WebApplet_InvokeMethod event.
2. Compile the project.
3. Start the Siebel application.
4. Navigate to the applet where the server script has been placed.
5. Perform an action on the applet that invokes a SWE method (for example, change the record or create a new record).

   The code generates a text file in the bin directory where the Siebel application is installed containing results of each method of the Web Engine HTTP TXN Business Service.

Sample Output

The following is an excerpt of the resulting text file.

```
====================================
WebApplet InvokeMethod event:
====================================

====================================
Method: GetAllRequestCookies
====================================
```

```
-------------------------------------
START LEVEL : 0
-------------------------------------
TYPE  : COOKIES
VALUE :
-------------------------------------
PROPERTIES
-------------------------------------
(NONE)
-------------------------------------
CHILD PROPERTY SETS
-------------------------------------

-------------------------------------
START LEVEL : 1
-------------------------------------
TYPE  : SWEUAID
VALUE : 1
-------------------------------------
PROPERTIES
-------------------------------------
Max-Age : -1
Domain :
Path :
-------------------------------------
CHILD PROPERTY SETS
-------------------------------------
(NONE)
-------------------------------------
END LEVEL : 1
-------------------------------------
-------------------------------------
END LEVEL : 0
-------------------------------------

=====================================
Method: GetAllRequestHeaders
=====================================

-------------------------------------
START LEVEL : 0
-------------------------------------
TYPE  : HEADERS
VALUE :
-------------------------------------
PROPERTIES
-------------------------------------
HOST : <host computer name>
CACHE-CONTROL : no-cache
CONNECTION : Keep-Alive
COOKIE : SWEUAID=1
USER-AGENT : Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; Q312461; SV1; .NET
CLR 1.1.4322)
CONTENT-TYPE : application/x-www-form-urlencoded
```

```
ACCEPT-ENCODING : deflate
CONTENT-LENGTH : 348
-------------------------------------
CHILD PROPERTY SETS
-------------------------------------
(NONE)
-------------------------------------
END LEVEL : 0
-------------------------------------

=====================================
Method: GetAllRequestParameters
=====================================

-------------------------------------
START LEVEL : 0
-------------------------------------
TYPE  : PARAMETERS
VALUE :
-------------------------------------
PROPERTIES
-------------------------------------
SWEActiveView : Account List View
SWERowIds :
SWEP :
SWESP : false
SWECmd : InvokeMethod
SWEMethod : PositionOnRow
SWER : 1
SWEControlClicked : 0
SWEIgnoreCtrlShift : 0
SWEVI :
SWEActiveApplet : Account List Applet
SWERPC : 1
SWEReqRowId : 1
SWEView : Account List View
SWEC : 3
SWERowId : 1-6
SWEShiftClicked : 0
SWETS : 1118939959734
SWEApplet : Account List Applet
-------------------------------------
CHILD PROPERTY SETS
-------------------------------------
(NONE)
-------------------------------------
END LEVEL : 0
-------------------------------------
```

Sample Code for Invoking Methods of Web Engine HTTP TXN Business Service

This topic contains the sample code for invoking the methods of the Web Engine HTTP TXN Business Service and writing the results to a text file. For more information, see "Example of Using Web Engine HTTP TXN Business Service" on page 111.

Add the following sample code to the WebApplet_InvokeMethod event:

```
function WebApplet_InvokeMethod (MethodName)
{
var fp = Clib.fopen('testfile.txt','a');
if ( fp == null )
{
TheApplication().RaiseErrorText(" ERROR Opening File ")
}
else
{
var oBS = TheApplication().GetService('Web Engine HTTP TXN');
var Inputs = TheApplication().NewPropertySet();
var Outputs = TheApplication().NewPropertySet();
var Headers = TheApplication().NewPropertySet();
var Cookies = TheApplication().NewPropertySet();
var tmpCookie = TheApplication().NewPropertySet();

Clib.fputs('====================================\n',fp);
Clib.fputs('WebApplet InvokeMethod event:\n',fp);
Clib.fputs('====================================\n',fp);

Clib.fputs('\n',fp);
Clib.fputs('====================================\n',fp);
Clib.fputs('Method: GetAllRequestCookies\n',fp);
Clib.fputs('====================================\n',fp);

Inputs.Reset();
Outputs.Reset();
oBS.InvokeMethod ( 'GetAllRequestCookies', Inputs, Outputs);
WebApplet_OutputPropertySet(Outputs, 0, fp);

Clib.fputs('\n',fp);
Clib.fputs('====================================\n',fp);
Clib.fputs('Method: GetAllRequestHeaders\n',fp);
Clib.fputs('====================================\n',fp);

Inputs.Reset();
Outputs.Reset();
oBS.InvokeMethod ('GetAllRequestHeaders', Inputs, Outputs);
WebApplet_OutputPropertySet(Outputs, 0, fp);

Clib.fputs('\n',fp);
Clib.fputs('====================================\n',fp);
Clib.fputs('Method: GetAllRequestParameters\n',fp);
Clib.fputs('====================================\n',fp);
```

```
Inputs.Reset();
Outputs.Reset();
oBS.InvokeMethod ('GetAllRequestParameters', Inputs, Outputs);
WebApplet_OutputPropertySet(Outputs, 0, fp);

Clib.fputs('\n',fp);
Clib.fputs('====================================\n',fp);
Clib.fputs('Method: GetAllResponseCookies\n',fp);
Clib.fputs('====================================\n',fp);

Inputs.Reset();
Outputs.Reset();
oBS.InvokeMethod ('GetAllResponseCookies', Inputs, Outputs)
WebApplet_OutputPropertySet(Outputs, 0, fp);

Clib.fputs('\n',fp);
Clib.fputs('====================================\n',fp);
Clib.fputs('Method: GetAllResponseHeaders\n',fp);
Clib.fputs('====================================\n',fp);

Inputs.Reset();
Outputs.Reset();
oBS.InvokeMethod ('GetAllResponseHeaders', Inputs, Outputs);
WebApplet_OutputPropertySet(Outputs, 0, fp);

Clib.fputs('\n',fp);
Clib.fputs('====================================\n',fp);
Clib.fputs('Method: GetAllServerVariables\n',fp);
Clib.fputs('====================================\n',fp);

Inputs.Reset();
Outputs.Reset();
oBS.InvokeMethod ('GetAllServerVariables', Inputs, Outputs);
WebApplet_OutputPropertySet(Outputs, 0, fp);

Clib.fputs('\n',fp);
Clib.fputs('====================================\n',fp);
Clib.fputs('Method: GetRequestCookies\n',fp);
Clib.fputs('====================================\n',fp);

Inputs.Reset();
Outputs.Reset();

Inputs.SetProperty ('MY-COOKIE', '');
Inputs.SetProperty ('TestCookie', '');
Inputs.SetProperty ('Test1Cookie', '');

oBS.InvokeMethod ('GetRequestCookies', Inputs, Outputs);
WebApplet_OutputPropertySet(Outputs, 0, fp);

Clib.fputs('\n',fp);
Clib.fputs('====================================\n',fp);
Clib.fputs('Method: GetRequestHeaders\n',fp);
Clib.fputs('====================================\n',fp);
```

```
Inputs.Reset();
Outputs.Reset();

Inputs.SetProperty ('MyHEADER', '');
Inputs.SetProperty ('MY_TEST', '');
Inputs.SetProperty ('CONTENT-TYPE', '');
Inputs.SetProperty ('CONTENT-LENGTH', '');

oBS.InvokeMethod ('GetRequestHeaders', Inputs, Outputs);
WebApplet_OutputPropertySet(Outputs, 0, fp);

Clib.fputs('\n',fp);
Clib.fputs('====================================\n',fp);
Clib.fputs('Method: GetRequestInfo\n',fp);
Clib.fputs('====================================\n',fp);

Inputs.Reset();
Outputs.Reset();

oBS.InvokeMethod ('GetRequestInfo', Inputs, Outputs);
WebApplet_OutputPropertySet(Outputs, 0, fp);

Clib.fputs('\n',fp);
Clib.fputs('====================================\n',fp);
Clib.fputs('Method: GetRequestParameters\n',fp);
Clib.fputs('====================================\n',fp);

Inputs.Reset();
Outputs.Reset();

Inputs.SetProperty ('TestQstr', '');
Inputs.SetProperty ('SWEActiveView', '');
Inputs.SetProperty ('SWECmd', '');
Inputs.SetProperty ('SWEMethod', '');
Inputs.SetProperty ('TestParam', '');

oBS.InvokeMethod ('GetRequestParameters', Inputs, Outputs);
WebApplet_OutputPropertySet(Outputs, 0, fp);

Clib.fputs('\n',fp);
Clib.fputs('====================================\n',fp);
Clib.fputs('Method: GetResponseCookies\n',fp);
Clib.fputs('====================================\n',fp);

Inputs.Reset();
Outputs.Reset();

Inputs.SetProperty ('My-Test-COOKIE', '');
Inputs.SetProperty ('_sn', '');

oBS.InvokeMethod ('GetResponseCookies', Inputs, Outputs);
WebApplet_OutputPropertySet(Outputs, 0, fp);
```

```
Clib.fputs('\n',fp);
Clib.fputs('====================================\n',fp);
Clib.fputs('Method: GetResponseHeaders\n',fp);
Clib.fputs('====================================\n',fp);

Inputs.Reset();
Outputs.Reset();

Inputs.SetProperty ('Content-Language', '');
Inputs.SetProperty ('MyHeader', '');

oBS.InvokeMethod ('GetResponseHeaders', Inputs, Outputs);
WebApplet_OutputPropertySet(Outputs, 0, fp);

Clib.fputs('\n',fp);
Clib.fputs('====================================\n',fp);
Clib.fputs('Method: GetResponseInfo\n',fp);
Clib.fputs('====================================\n',fp);

Inputs.Reset();
Outputs.Reset();

oBS.InvokeMethod ('GetResponseInfo', Inputs, Outputs);
WebApplet_OutputPropertySet(Outputs, 0, fp);

Clib.fputs('\n',fp);
Clib.fputs('====================================\n',fp);
Clib.fputs('Method: GetServerVariables\n',fp);
Clib.fputs('====================================\n',fp);

Inputs.Reset();
Outputs.Reset();

Inputs.SetProperty ('AUTH-USER-ID', '');
Inputs.SetProperty ('SERVER-NAME', '');

oBS.InvokeMethod ('GetServerVariables', Inputs, Outputs);
WebApplet_OutputPropertySet(Outputs, 0, fp);

Clib.fputs('\n',fp);
Clib.fputs('====================================\n',fp);
Clib.fputs('Method: GetWebSessionInfo\n',fp);
Clib.fputs('====================================\n',fp);

Inputs.Reset();
Outputs.Reset();

oBS.InvokeMethod ('GetWebSessionInfo', Inputs, Outputs);
WebApplet_OutputPropertySet(Outputs, 0, fp);

Clib.fputs('\n',fp);
Clib.fputs('====================================\n',fp);
Clib.fputs('Method: SetResponseCookies\n',fp);
Clib.fputs('====================================\n',fp);
```

```
Inputs.Reset();
Outputs.Reset();

tmpCookie = null;
tmpCookie = TheApplication().NewPropertySet();

tmpCookie.SetType ('My_Test_Cookie');
tmpCookie.SetValue ('Cookie Value for My_Test_Cookie');
tmpCookie.SetProperty ('Max-Age', '23434343');
tmpCookie.SetProperty ('Domain', '.example.com');
tmpCookie.SetProperty ( 'Path', 'eapps/test/cookie/path');

Inputs.AddChild (tmpCookie);

tmpCookie = null;
tmpCookie = TheApplication().NewPropertySet();

tmpCookie.SetType ('Another_Cookie');
tmpCookie.SetValue ('Cookie Value for Another_Cookie');
tmpCookie.SetProperty ('Max-Age', '23434343');
tmpCookie.SetProperty ('Domain', 'esales.example.com');
tmpCookie.SetProperty ('Path', 'esales/cookie/path');

Inputs.AddChild (tmpCookie);

oBS.InvokeMethod ('SetResponseCookies', Inputs, Outputs);
Clib.fputs('-----------------------------------\n',fp);
Clib.fputs('Input Cookies\n',fp);
Clib.fputs('-----------------------------------\n',fp);
WebApplet_OutputPropertySet(Inputs, 0, fp);

oBS.InvokeMethod ('GetAllResponseCookies', Inputs, Outputs);
Clib.fputs('-----------------------------------\n',fp);
Clib.fputs('Output Cookies\n',fp);
Clib.fputs('-----------------------------------\n',fp);
WebApplet_OutputPropertySet(Outputs, 0, fp);

Clib.fputs('\n',fp);
Clib.fputs('===================================\n',fp);
Clib.fputs('Method: SetResponseHeaders\n',fp);
Clib.fputs('===================================\n',fp);

Inputs.Reset();
Outputs.Reset();

Inputs.SetProperty ('MyHeader', 'THIS is MyHeader');

oBS.InvokeMethod ('SetResponseHeaders', Inputs, Outputs);
Clib.fputs('-----------------------------------\n',fp);
Clib.fputs('Input Headers\n',fp);
Clib.fputs('-----------------------------------\n',fp);
WebApplet_OutputPropertySet(Inputs, 0, fp)
```

```
oBS.InvokeMethod ('GetAllResponseHeaders', Inputs, Outputs);
Clib.fputs('-----------------------------------\n',fp);
Clib.fputs('Output Headers\n',fp);
Clib.fputs('-----------------------------------\n',fp);
WebApplet_OutputPropertySet(Outputs, 0, fp);

Clib.fputs('\n',fp);
Clib.fputs('===================================\n',fp);
Clib.fputs('Method: SetResponseInfo\n',fp);
Clib.fputs('===================================\n',fp);

Inputs.Reset();
Outputs.Reset();
Headers.Reset();
Cookies.Reset();

Headers.SetType ('HEADERS');
Headers.SetProperty ('ABC_RESPONSE_HEADER1', 'RESPONSE_HEADER1 Value');
Headers.SetProperty ('ABC_RESPONSE_HEADER2', 'RESPONSE_HEADER2 Value');
Headers.SetProperty ('ABC_RESPONSE_HEADER3', 'RESPONSE_HEADER3 Value');
Headers.SetProperty ('ABC_RESPONSE_HEADER4', 'RESPONSE_HEADER4 Value');
Inputs.AddChild( Headers);

Cookies.SetType('COOKIES');

tmpCookie = null;
tmpCookie = TheApplication().NewPropertySet();

tmpCookie.SetType ('My_Test_Cookie2');
tmpCookie.SetValue ( 'Cookie Value for My_Test_Cookie2');
tmpCookie.SetProperty ( 'Max-Age', '23434343');

Cookies.AddChild (tmpCookie);

tmpCookie = null;
tmpCookie = TheApplication().NewPropertySet();

tmpCookie.SetType ('Another_Cookie2');
tmpCookie.SetValue ('Cookie Value for Another_Cookie2');
tmpCookie.SetProperty ('Max-Age', '23434343');

Cookies.AddChild (tmpCookie);

Inputs.AddChild (Cookies);

oBS.InvokeMethod ('SetResponseInfo', Inputs, Outputs);
Clib.fputs('-----------------------------------\n',fp);
Clib.fputs('Input Info\n',fp);
Clib.fputs('-----------------------------------\n',fp);
WebApplet_OutputPropertySet(Inputs, 0, fp);

oBS.InvokeMethod ('GetResponseInfo', Inputs, Outputs);
Clib.fputs('-----------------------------------\n',fp);
Clib.fputs('Output Info\n',fp);
Clib.fputs('-----------------------------------\n',fp);
WebApplet_OutputPropertySet(Outputs, 0, fp);
```

```
Clib.fclose(fp);
}
}
```

Index

Q

R

S