CONVERSANT® System
Version 8.0
Application Development with
Siebel eBusiness
# Contents

## Introduction
- Prerequisites .................................................. v
- Basic Concepts ............................................... vi
- Creating an Application .................................... vii

## Identifying Data to be Transferred
- Data on Siebel eBusiness ..................................... 1
- Data on CONVERSANT ........................................ 2

## Siebel Setup
- Siebel Overview .............................................. 3
- Integration Objects .......................................... 4
- Workflows ....................................................... 5
- Document Type Definitions .................................. 6

## XML Templates
- Overview .......................................................... 9
- Interpreting DTDs .............................................. 10
- Creating the Template ....................................... 12

## Queries and Updates
- Overview .......................................................... 13
- The Vonetix Interface ........................................ 13
- Performing Queries and Updates ............................. 15

## Sample Application
- Overview .......................................................... 17
- XML Documents ................................................ 20
  - Query .......................................................... 20
  - Query Returned ............................................. 20
  - Update .......................................................... 21
  - Update Returned ............................................ 21
Contents

Voice@Work Application Detail ........................................... 22
Query ................................................................. 23
Setting Up a Transfer ...................................................... 26
Updating Siebel eBusiness to Perform a Transfer .............. 30

Appendix A: References

Siebel eBusiness .......................................................... 35
CONVERSANT ............................................................. 35
Introduction

This document describes how to create applications for CONVERSANT System Version 8.0 that can exchange data with Siebel eBusiness products.

Prerequisites

The following assumes:

- Familiarity with Siebel eBusiness (setup, administration, and use).
- Familiarity with CONVERSANT application development, preferably using Voice@Work.
- Familiarity with the Extensible Markup Language (XML).

Any CONVERSANT system running an application to interface with Siebel eBusiness needs to have Gold Systems Vonetix version 2.1 or higher loaded on the system.
Basic Concepts

As shown in the figure "CONVERSANT and Siebel eBusiness" on page vi, applications on CONVERSANT exchange information with Siebel eBusiness databases using XML documents. This is facilitated by Vonetix middleware on CONVERSANT and the Business Integration Manager on Siebel eBusiness.

CONVERSANT and Siebel eBusiness

Vonetix enables the application to read, create, and send XML documents from CONVERSANT. The Business Integration Manager manages the translation and transfer of XML documents within Siebel eBusiness, and accesses Siebel eBusiness databases using abstract objects that encapsulate the data of interest.
Creating an Application

There are several basic steps involved in creating an application on CONVERSANT to access and/or send data to a Siebel eBusiness database. These steps provide the basic organization of this guide for application developers:

1. Identify which data will be transferred from Siebel eBusiness to CONVERSANT. See Chapter 1: Identifying Data to be Transferred on page 1.

2. Identify which data will be transferred from CONVERSANT to Siebel eBusiness. See Chapter 1: Identifying Data to be Transferred on page 1.

3. Create a Siebel integration object that represents the data of interest. See Chapter 2: Siebel Setup on page 3.

4. Create a workflow within Siebel eBusiness that manages the exchange of information between Siebel eBusiness and CONVERSANT. See Chapter 2: Siebel Setup on page 3.

5. Use Siebel eBusiness to generate a document type definition (DTD) for each integration object. See Chapter 2: Siebel Setup on page 3.

6. Use the DTD (from Step 5) to create an XML template for files that will be sent from CONVERSANT to Siebel eBusiness. See Chapter 3: XML Templates on page 9.

7. Create the CONVERSANT application using either Voice@Work or Script Builder. The application uses the XML template to generate an XML document which is sent to Siebel eBusiness. See Chapter 4: Queries and Updates on page 13.

A sample application is provided at the end of the document. See Chapter 5: Sample Application on page 17.
Chapter 1 Identifying Data to be Transferred

An application on CONVERSANT will be doing one or both of the following:

- Querying Siebel eBusiness for information in a Siebel database (retrieve data from Siebel eBusiness).
- Updating a Siebel eBusiness database (send data from CONVERSANT to Siebel eBusiness).

The first task for the application developer is to decide what data is of interest, and where it is.

Data on Siebel eBusiness

Data on Siebel eBusiness resides in standard database files, organized according to what Siebel calls the “Siebel Data Model”.

The data is accessible through object-oriented software. The software enables a user to group related data from various parts of the database into a “business object” which can be treated as a unit.

EXAMPLE:

Names, occupations, and ages could be manipulated together as a “People” object. To get someone’s name, the application would need to access the People object.

Data of interest may reside in several business objects.

When creating an application, it is important to identify the business object(s) that have data that will be queried and data that will be updated.
Data on CONVERSANT

Data on CONVERSANT can be

- Collected directly from a caller (in a “prompt and collect” operation).
- Associated with the call (such as ANI or DNIS).
- Stored in a database.
- In a file.
Chapter 2  Siebel Setup

Siebel Overview

To access business objects, Siebel's Business Integration Manager uses a Siebel EAI (eBusiness Application Interface) Adapter, which translates business objects or some of their components into "integration objects" that can be used to create XML documents. Information can also go the other way: integration objects can be translated into business objects or some of their components through the EAI Adapter. Note that parts of several different business objects can be included in a single integration object.

The management of data within Siebel eBusiness is done using "workflows", which are customizable processes and rules. They control access, assembly, and distribution of integration objects. A workflow can deal with queries, updates, or both.

XML documents are exchanged with external applications (on a CONVERSANT) via HTTP using the Business Integration Manager's XML Gateway and HTTP Gateway.

See the Siebel 2000 Bookshelf for more information.
Integration Objects

Integration objects, like business objects, are made of components, which themselves are made of fields, as shown in the figure "Part of a Typical Integration Object" on page 4. This is comparable to the parent-child-data relationships found in XML.

Siebel eBusiness has the ability to create integration objects using the Integration Object Builder through Siebel Tools (see the Siebel 2000 Bookshelf: Siebel Integration Guide for instructions).

Part of a Typical Integration Object
Workflows

Workflows need to be created to enable the proper operations to be done within Siebel eBusiness. One or more workflows may be used, depending on the customer’s needs.

A sample workflow for a query is shown in the figure "Sample Workflow (Query)" on page 5. In this workflow, an XML document is received from CONVERSANT, a Siebel database is queried, and an XML document is prepared and sent to CONVERSANT.

Sample Workflow (Query)

The figure "Sample Workflow (Update)" on page 5 shows a sample workflow for an update. In this case, data is inserted into a Siebel database after Siebel eBusiness receives an XML document from CONVERSANT. An XML document is prepared and sent to CONVERSANT.

Sample Workflow (Update)

The Siebel Workflow Guide in the Siebel Bookshelf describes how to set up workflows.
Document Type Definitions

XML files are generally used to transport data across networks using Hypertext Transfer Protocol (HTTP). Document type definitions (DTDs) provide information about how XML documents are formed.

Siebel eBusiness has the equivalent of a DTD for every integration object, which it uses to create outgoing XML documents and to decide whether incoming XML documents have the proper format (the documents are “validated” by Siebel eBusiness). CONVERSANT does not use DTDs at all, but DTDs generated by Siebel eBusiness can be used to construct CONVERSANT applications which will be able to send and interpret XML files.

To create a DTD, generate a schema for the integration object within Siebel eBusiness.

In the examples below, separate integration objects exist for query and update operations. Note that this may be the case more often than having one DTD for both operations:

Sample Query DTD

```xml
<!DOCTYPE BubbaAccountDetail (FincorpAccount+) >
<!ELEMENT BubbaAccountDetail (FincorpAccount+) >
<!ELEMENT FincorpAccount (AccountNumber?,
RelationshipLimit?,
CurrentBalance?,
LastName?)>
<!ELEMENT AccountNumber (#PCDATA) >
<!ELEMENT RelationshipLimit (#PCDATA) >
<!ELEMENT CurrentBalance (#PCDATA) >
<!ELEMENT LastName (#PCDATA) >
```
Sample Update DTD

<!-- Copyright (C) 1994, Siebel Systems, L.P., All rights reserved. -->
<!-- Siebel DTD Generation -->
<!ELEMENT BubbaFundsTransferSr (FundsTransferSR+) >
<!ELEMENT FundsTransferSR (ContactFinancialAccounts?,
    ContactLastName?,
    FundsTransferFromAccountNumber?,
    FundsTransferToAccountNumber?,
    FundsTransferDollarAmount?,
    TransactionId?,
    SRNumber?)>  
<!ELEMENT ContactFinancialAccounts (#PCDATA) >
<!ELEMENT ContactLastName (#PCDATA) >
<!ELEMENT FundsTransferFromAccountNumber (#PCDATA) >
<!ELEMENT FundsTransferToAccountNumber (#PCDATA) >
<!ELEMENT FundsTransferDollarAmount (#PCDATA) >
<!ELEMENT TransactionId (#PCDATA) >
<!ELEMENT SRNumber (#PCDATA) >

The DTD is used by the application differently for reading incoming documents and creating outgoing documents. Every XML document must match the element definitions given in the DTD.

- For XML documents being received by CONVERSANT, the element names need to be known so that the application can parse the document and extract the data.

- For XML documents being sent by CONVERSANT, an XML template can be created separately by the developer, which is then used to build the XML documents that will be sent. The documents are always populated with data. For queries, this data will typically be a key field. Updates include data that will be put into a Siebel database.
Chapter 3  XML Templates

Overview

XML documents sent from CONVERSANT to Siebel eBusiness, whether they be for queries or updates, can be constructed using XML templates that derive from the DTDs supplied by Siebel eBusiness for the related integration objects.

As mentioned in Chapter 2: Siebel Setup on page 3, a DTD provides the application developer with the structure of the XML document that Siebel eBusiness expects. Of principal importance are:

- Element names.
- Element order.
- Data types.

A template has two main features:

- A header, which tells Siebel eBusiness which integration object is being addressed.
- A body of tag pairs, each having either no content or reusable content.
Interpreting DTDs

Following is a brief discussion of how to interpret DTDs.

The structure of a DTD can be seen by example:

```xml
<!-- Copyright (C) 1994, Siebel Systems, L.P., All rights reserved. -->
<!-- Siebel DTD Generation -->
<!ELEMENT BubbaFundsTransferSr (FundsTransferSR+) >
<!ELEMENT FundsTransferSR (ContactFinancialAccounts?,
    ContactLastName?,
    FundsTransferFromAccountNumber?,
    FundsTransferToAccountNumber?,
    FundsTransferDollarAmount?,
    TransactionId?,
    SRNumber?)>
<!ELEMENT ContactFinancialAccounts (#PCDATA) >
<!ELEMENT ContactLastName (#PCDATA) >
<!ELEMENT FundsTransferFromAccountNumber (#PCDATA) >
<!ELEMENT FundsTransferToAccountNumber (#PCDATA) >
<!ELEMENT FundsTransferDollarAmount (#PCDATA) >
<!ELEMENT TransactionId (#PCDATA) >
<!ELEMENT SRNumber (#PCDATA) >
```

The first two lines

```xml
<!-- Copyright (C) 1994, Siebel Systems, L.P., All rights reserved. -->
<!-- Siebel DTD Generation -->
```

are the header. The Siebel header will not be used in the XML template.

The remainder of the DTD defines the elements expected in the body of the XML document. Each line is called an “element declaration”, as identified by the `!ELEMENT` label.

An element declaration typically has two parts:

1. The name of the element, identified by a tag.
2. A description of the element’s contents (enclosed in parentheses).

In the example,

```xml
<!ELEMENT SRNumber (#PCDATA) >
```

`SRNumber` is the element name, and `#PCDATA` describes the contents as parsed character data.
Slightly more complicated is the line

```xml
<!ELEMENT FundsTransferSR (ContactFinancialAccounts?,
    ContactLastName?,
    FundsTransferFromAccountNumber?,
    FundsTransferToAccountNumber?,
    FundsTransferDollarAmount?,
    TransactionId?,
    SRNumber?)>
```

In this case, the contents of the element are a set of child elements, where each child’s name is separated by a comma. Note that a “?”, “*”, or “+” following a child name indicates the number of instances of the element that are allowed (zero or more instances can be present between tags for “?” or “*”; zero or more instances must be present for “+”).
Creating the Template

The XML template will need to be created and stored as a file on the CONVERSANT. A header naming the integration object will need to be added to the tag pairs (start and end tags) derived from the DTD. The header takes the form

```
<SiebelMessage MessageID=""
MessageType="IntegrationObject"
IntObjectName="IntegObjName">
```

Where `IntegObjName` is the name of the integration object.

The DTD example shown above would be translated into the following template:

```
<SiebelMessage MessageID=""
MessageType="IntegrationObject"
IntObjectName="IntegObjName">
<BubbaFundsTransferSr>
<FundsTransferSR>
<ContactFinancialAccounts></ContactFinancialAccounts>
<ContactLastName></ContactLastName>
<FundsTransferFromAccountNumber></FundsTransferFromAccountNumber>
<FundsTransferToAccountNumber></FundsTransferToAccountNumber>
<FundsTransferDollarAmount></FundsTransferDollarAmount>
<TransactionId></TransactionId>
<SRNumber></SRNumber>
</FundsTransferSR>
</BubbaFundsTransferSr>
</SiebelMessage>
```

When using the template to update a Siebel eBusiness database, the CONVERSANT application will populate the elements with data.
Chapter 4  Queries and Updates

Overview

CONVERSANT communicates with Siebel eBusiness using a plug-in that is part of the Vonetix middleware by Gold Systems. An application on CONVERSANT can make function calls to Vonetix that

- Prepare and send outgoing XML documents.
- Receive and process incoming XML documents.

What Siebel eBusiness actually receives from CONVERSANT is a set of “form data” that includes the original XML document on CONVERSANT along with special information that tells Siebel eBusiness what to do.

The Vonetix Interface

Information flow between an application and Siebel eBusiness is facilitated by specialized Vonetix functionality, which is illustrated in the figure “CONVERSANT and the Vonetix Interface” on page 13.
The Vonetix interface consists of the following parts:

- **Java Gateway DIP.**
  Applications communicate directly with this data interface process, which uses TCP/IP to pass information to the Vonetix Gateway.

- **Vonetix Gateway.**
  This gateway sends requests to the Mini-Browser based upon information received from the DIP.

- **Mini-Browser.**
  The Mini-Browser is where XML documents are interrogated and modified, and HTTP requests are sent to the Base Servlet.

- **Base Servlet.**
  The Base Servlet receives HTTP requests from the Mini-Browser via the Web Server and Tomcat Servlet, then relays them to the Document Plug-In, tracking them by unique session names and cookies.

- **Document Plug-In.**
  The Document Plug-In relays HTTP requests and XML documents between the Base Servlet and Siebel eBusiness.

Vonetix provides TAS functions to support the Document Plug-In, the “Generic TAS Functions” and “Utilities TAS Functions” (see the latest Vonetix *Technical Training Guide* or *Data Channel Guide* for a detailed listing with descriptions and code examples).
Performing Queries and Updates

There are basic steps performed by a CONVERSANT application for queries and updates.

Note that the Vonetix functions discussed in the steps below, unless otherwise mentioned, will return a 0 if successful, or a negative number if unsuccessful. See the Vonetix Return Code Table in the applicable Vonetix Technical Training Guide or Data Channel Guide.

1. Prepare the original XML document.
   a. Load the XML template.
      Use the Vonetix \texttt{vntxLoadDoc(SESSIONNAME, FILENAME)} function, where the argument \texttt{FILENAME} includes the path to the template from the root directory.
   b. Add content to one or more elements in the document.
      This involves using the Vonetix \texttt{vntxSetTag(SESSIONNAME, TAGNAME, TAGNUMBER, TAGVALUE)} function for each element identified by the arguments \texttt{TAGNAME} and \texttt{TAGNUMBER}. The content of the element is substituted for \texttt{TAGVALUE}.

2. Prepare form data for the HTTP request.
   HTTP request form data is used to log on to Siebel eBusiness, identify the XML document, and tell Siebel eBusiness what to do.
   This form data consists of one or more fields that are logically tied together by a session name, and are each created using the Vonetix \texttt{vntxSetFD(SESSIONNAME, FIELDNAME, FIELDVALUE)} function, where the argument \texttt{FIELDNAME} is the field's name, and the argument \texttt{FIELDVALUE} is the field's value.
   Each of the following fields needs to be defined:
   a. \texttt{SWEExtSource}
      The workflow name (Business Service Source), found in the Siebel eBusiness config file.
   b. \texttt{SWEExtCmd}
      The command to be executed by Siebel eBusiness. The most commonly used command is likely to be \texttt{Execute}.
   c. \texttt{UserName}
      The login name.
   d. \texttt{Password}
      The login password.
3. Convert the original XML document into an encoded form data value.

   The XML document must itself be turned into form data before being sent by Vonetix. The Vonetix 
   \texttt{VntxSetDocFD(SESSIONNAME, name)} function does this, where \texttt{name} is “SWEExtData”.

4. Send the form data.

   The form data sent to Siebel eBusiness consists of the XML document created from the XML template, coupled with the HTTP request form data that was created in Step 2.

   Use the Vonetix \texttt{vntxPostDoc(SESSIONNAME, URL)} function, where the argument \texttt{URL} is the URL of the Website with the Siebel eBusiness data.

   An error code will be returned by Vonetix if Siebel eBusiness has not responded within 20 seconds of the HTTP request.

5. Save the incoming XML document. (This step is optional, usually for debugging purposes.)

   Siebel eBusiness will respond to the POST command by sending a document to CONVERSANT. The received document can be saved regardless of whether the CONVERSANT application is doing a query or an update.

   Use the Vonetix \texttt{vntxStoreDoc(SESSIONNAME, FILENAME)} function, where the argument \texttt{FILENAME} refers to the path and name you choose for the file that will be stored.

6. Process the received XML document.

   Extract the content of each element of interest. This will certainly be done during a query, but may also be done after an update.

   Use the Vonetix \texttt{vntxGetTag(SESSIONNAME, TAGNAME, TAGNUMBER, TAGTEXT)} function. \texttt{TAGTEXT} is the name of the variable that will store the content.

7. Clear form data.

   After performing a query or update, the form data encoded XML document and/or HTTP request form data may need to be cleared. This action will not remove data saved in variables and stored XML documents.

   - Use the \texttt{vntxClearSes(SESSIONNAME)} function to clear \textit{both} the XML document and HTTP request form data.
   - Use the \texttt{vntxClearDoc(SESSIONNAME)} function to clear just the form data encoded XML document.
   - Use the \texttt{vntxClearFD(SESSIONNAME)} function to clear just the HTTP request form data.
Chapter 5  Sample Application

Overview

In this chapter, a sample application is described that includes both a query and an update of information in Siebel eBusiness.

The application deals with the transfer of money from one bank account to another. See Sample Application Flowchart - 1 on page 18 and Sample Application Flowchart - 2 on page 19.

1. The application greets the caller.
2. The caller is taken to the main menu.
3. The main menu prompts the caller to select a type of account.
4. The caller is prompted for the account number of the account chosen in Step 3.
5. The application queries Siebel eBusiness for the account balance and the name of the account’s owner.
   - If an error is received from Siebel eBusiness, then the caller is informed and returned to the main menu. The caller can have a maximum of three errors, after which the application terminates.
   - The application speaks the account balance and name on the account.
6. The second menu prompts the caller for the type of account to transfer money from and the type of account to transfer money to.
7. The caller is prompted to enter the account numbers for the two account types chosen in Step 6.
8. The caller is prompted to enter the amount of money to be transferred.
9. Siebel eBusiness is queried for the validity of the account numbers entered in Step 8.
   Any errors are reported to the caller, who is then routed back to the main menu.
10. The caller is asked to confirm the transfer.
    If the transfer is not confirmed, then the caller is routed to the main menu.
11. The application updates the Siebel eBusiness database. If there is an error, then the caller is informed of the error and then routed to the main menu.

12. The application speaks the transaction code to the caller.

13. The caller is routed to the main menu.

Sample Application Flowchart - 1
XML Documents

In the documents below, the account number has been included as a “key” for accessing the customer’s information.

Query

```xml
<SiebelMessage MessageId=""
MessageType="Integration Object"
IntObjectName="Bubba Account Detail">
<BubbaAccountDetail>
<FincorpAccount>
<AccountNumber>999-999-999</AccountNumber>
</FincorpAccount>
</BubbaAccountDetail>
</SiebelMessage>
```

Query Returned

```xml
<?xml version="1.0" encoding="UTF-8"?>
<SiebelMessage IntObjectName="Bubba Account Detail" MessageId=""
MessageType="Integration Object">
<BubbaAccountDetail>
<FincorpAccount>
<AccountNumber>999-999-999</AccountNumber>
<RelationshipLimit>10000</RelationshipLimit>
<CurrentBalance>200000</CurrentBalance>
<LastName>Jordan</LastName>
</FincorpAccount>
</BubbaAccountDetail>
</SiebelMessage>
```
Update

<?xml version="1.0" encoding="UTF-8"?>
<SiebelMessage IntObjectName="Bubba Funds Transfer SR" MessageId="" MessageType="Integration Object">
  <BubbaFundsTransferSr>
    <FundsTransferSR>
      <ContactFinancialAccounts>999-999-999</ContactFinancialAccounts>
      <ContactLastName>Jordan</ContactLastName>
      <FundsTransferFromAccountNumber>111-111-111</FundsTransferFromAccountNumber>
      <FundsTransferToAccountNumber>555-555-555</FundsTransferToAccountNumber>
      <FundsTransferDollarAmount>20000</FundsTransferDollarAmount>
      <TransactionId>IVR-Transfer-3</TransactionId>
    </FundsTransferSR>
  </BubbaFundsTransferSr>
</SiebelMessage>

Update Returned

<?xml version="1.0" encoding="UTF-8"?>
<SiebelMessage IntObjectName="Bubba Funds Transfer SR" MessageId="" MessageType="Integration Object">
  <BubbaFundsTransferSr>
    <FundsTransferSR>
      <ContactFinancialAccounts>999-999-999</ContactFinancialAccounts>
      <ContactLastName>Jordan</ContactLastName>
      <FundsTransferFromAccountNumber>111-111-111</FundsTransferFromAccountNumber>
      <FundsTransferToAccountNumber>555-555-555</FundsTransferToAccountNumber>
      <FundsTransferDollarAmount>20000</FundsTransferDollarAmount>
      <TransactionId>IVR-Transfer-3</TransactionId>
    </FundsTransferSR>
  </BubbaFundsTransferSr>
</SiebelMessage>
Voice@Work Application Detail

Following is Voice@Work output that shows parts of the sample application. This discussion will focus on what is relevant to the interactions with Siebel eBusiness.

The figure below ("Initial Steps" on page 22) shows the first set of nodes for the application. As described in the Overview on page 17, the application first answers the call and prompts the caller for the type of account in the main menu.

Initial Steps

PCAccNum (see "PCAccNum Detail" on page 23) sets up and executes both a balance query and the transfer of funds from one account to another.
At the beginning of PCAccNum, Siebel eBusiness is queried for the account balance and account owner's name (in CFCheckAccNum). DBTimeStamp is a debugging step. These steps are shown in the figure "CFCheckAccNum Detail" on page 24.
The XML template is loaded (\texttt{VntxLoadDoc2}), and is used as the outgoing XML document after having its \texttt{AccountNumber} element populated (in \texttt{VntxSetTag1}) with the account number that Siebel eBusiness will use to look up the information wanted.
The `VntxSetFD` function is used to add form data in nodes `VntxSetFD1` (for `SWEExtSource`), `VntxSetFD2` (for `SWEExtCmd`), `VntxSetFD3` (for `UserName`), and `VntxSetFD4` (for `Password`).

The XML document is then converted into form data in node `VntxSetDocFD2`, and POSTed to Siebel eBusiness in node `VntxPostDoc2`.

A document is sent by Siebel eBusiness according to its workflow, and processed via `VntxStoreDoc1` as shown in "VntxStoreDoc1 Detail" on page 25.

**VntxStoreDoc1 Detail**
Sample Application

VntxStoreDoc1 uses the VntxStoreDoc function to store the XML document sent from Siebel eBusiness. Elements of interest are then read from the XML document using the VntxGetTag function (this function name is part of the relevant node names in the figure above). After some manipulation of the information that was read, the HTTP request form data is cleared (in VntxClearFD1), and then the form data encoded XML document is also cleared (in VntxClearDoc1).

<table>
<thead>
<tr>
<th>Node</th>
<th>Function</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>VntxStoreDoc1</td>
<td>VntxStoreDoc</td>
<td>SNAME=&quot;firstacc&quot;, FNAME=&quot;/home/bankdemo/out/accmaster.out&quot;</td>
</tr>
<tr>
<td>VntxGetTagLimit</td>
<td>VntxGetTag</td>
<td>SNAME=&quot;firstacc&quot;, TNAME=&quot;RelationShipLimit&quot;, TNUMB=1, ATEXT=[MainAccLimit]</td>
</tr>
<tr>
<td>VntxGetTagCName</td>
<td>VntxGetTag</td>
<td>SNAME=&quot;firstacc&quot;, TNAME=&quot;LastName&quot;, TNUMB=1, ATEXT=[ContactLastName]</td>
</tr>
<tr>
<td>VntxGetTagAccNum</td>
<td>VntxGetTag</td>
<td>SNAME=&quot;firstacc&quot;, TNAME=&quot;CurrentBalance&quot;, TNUMB=1, ATEXT=[CurrBalance]</td>
</tr>
<tr>
<td>VntxClearFD1</td>
<td>VntxClearFD</td>
<td>SESS=&quot;firstacc&quot;</td>
</tr>
<tr>
<td>VntxClearDoc1</td>
<td>VntxClearDoc</td>
<td>SESS=&quot;firstacc&quot;</td>
</tr>
</tbody>
</table>

Setting Up a Transfer

The second menu that the caller encounters is covered in the next part of PCAccNum (see “Second Menu Detail” on page 27). In this part the caller is prompted for the accounts to be used in the transfer. The choices of the accounts are based upon the account type that the caller chose in the main menu:

- **PCMenuLoan** is entered if the account type was “Loan”.
- **PCFundMenu** is entered if the account type was “Fund”.
- **PCSsaveMenu** is entered if the account type was “Save”.

26 CONVERSANT System Version 8.0 Application Development with Siebel eBusiness
Second Menu Detail

**PCSsaveMenu** is typical of the process used for prompting the caller for accounts and for transferring funds, as shown in "PCSsaveMenu Detail (from PCAccNum)” on page 27:
The caller selects which account money will be transferred from, and which account the money will be transferred to. The application then branches to the appropriate call flow (*SetSVFromTo2*, *SetSVFromTo3*, or *SetSVFromTo1*) and performs the transfer (*CFPayment*).

*CFPayment* starts with two account number queries (see "CFPayment Detail" on page 29). *CFCheckFromAccNum* queries Siebel eBusiness for the number of the account that money will be taken from, and *CFCheckToAccNum* queries Siebel eBusiness for the number of the account that money will be transferred to. The form of these routines is similar to that of *CFCheckAccNum* (see CFCheckAccNum Detail on page 24).
The actual transfer of money, in the form of a Siebel eBusiness update, occurs in **PCConfirm** (see "PCConfirm Detail" on page 30).
Updating Siebel eBusiness to Perform a Transfer

CFUpdate has the update steps, as shown in "CFUpdate Detail" on page 31.
CFUpdate Detail

SetToVar

VmxLoadList

VmxSetTag2

VmxSetTag4

SetSelTime

VmxSetTag5

VmxSetFD2

VmxSetFD3

VmxSetFD4

VmxSetDocFD2

SetAndTest1

VmxPostDoc2

SetAndTest2

UBTLastStamp

VmxStoreDoc3

VmxStoreDoc3

VmxSetTag1

VmxClearFD1

VmxClearDoc1

TestReturn

(GoodLookup) (Return)

(default)
<table>
<thead>
<tr>
<th>Node</th>
<th>Function</th>
<th>Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>VntxLoadLast</td>
<td>VntxLoadDoc</td>
<td>SESS=&quot;last&quot;, FILE=&quot;/home/bankdemo/accUpdate.xml&quot;</td>
</tr>
<tr>
<td>VntxSetTag8</td>
<td>VntxSetTag</td>
<td>SNAME=&quot;last&quot;, TNAME=&quot;ContactFinancialAccounts&quot;, TNUMB=1, VALUE=[AccNum]</td>
</tr>
<tr>
<td>VntxSetTag1</td>
<td>VntxSetTag</td>
<td>SNAME=&quot;last&quot;, TNAME=&quot;ContactLastName&quot;, TNUMB=1, VALUE=[ContactLastName]</td>
</tr>
<tr>
<td>VntxSetTag2</td>
<td>VntxSetTag</td>
<td>SNAME=&quot;last&quot;, TNAME=&quot;FundsTransferFromAccountNumber&quot;, TNUMB=1, VALUE=[FAccNum]</td>
</tr>
<tr>
<td>VntxSetTag3</td>
<td>VntxSetTag</td>
<td>SNAME=&quot;last&quot;, TNAME=&quot;FundsTransferToAccountNumber&quot;, TNUMB=1, VALUE=[TAccNum]</td>
</tr>
<tr>
<td>VntxSetTag4</td>
<td>VntxSetTag</td>
<td>SNAME=&quot;last&quot;, TNAME=&quot;FundsTransferDollarAmount&quot;, TNUMB=1, VALUE=[TransAmount]</td>
</tr>
<tr>
<td>VntxSetTag5</td>
<td>VntxSetTag</td>
<td>SNAME=&quot;last&quot;, TNAME=&quot;TransactionID&quot;, TNUMB=1, VALUE=[TransID]</td>
</tr>
<tr>
<td>VntxSetFD1</td>
<td>VntxSetFD</td>
<td>SESS=&quot;last&quot;, NAME=&quot;SWEExtSource&quot;, TEXT=&quot;xmlupdate&quot;</td>
</tr>
<tr>
<td>VntxSetFD2</td>
<td>VntxSetFD</td>
<td>SESS=&quot;last&quot;, NAME=&quot;SWEExtCmd&quot;, TEXT=&quot;Execute&quot;</td>
</tr>
<tr>
<td>VntxSetFD3</td>
<td>VntxSetFD</td>
<td>SESS=&quot;last&quot;, NAME=&quot;UserName&quot;, TEXT=&quot;SADMIN&quot;</td>
</tr>
<tr>
<td>VntxSetFD4</td>
<td>VntxSetFD</td>
<td>SESS=&quot;last&quot;, NAME=&quot;Password&quot;, TEXT=&quot;SADMIN&quot;</td>
</tr>
<tr>
<td>VntxSetDocFD2</td>
<td>VntxSetDocFD</td>
<td>SESS=&quot;last&quot;, DOC=&quot;SWEExtData&quot;</td>
</tr>
<tr>
<td>VntxPostDoc2</td>
<td>VntxPostDoc</td>
<td>SESS=&quot;last&quot;, URL=&quot;http://.../eai/start.swe&quot;</td>
</tr>
<tr>
<td>VntxStoreDoc3</td>
<td>VntxStoreDoc</td>
<td>SNAME=&quot;last&quot;, FNAME=&quot;/home/bankdemo/out/accUpdate.out&quot;</td>
</tr>
<tr>
<td>VntxGetTag1</td>
<td>VntxGetTag</td>
<td>SNAME=&quot;last&quot;, TNAME=&quot;SRNumber&quot;, TNUMB=1, ATEXT=[SRNum]</td>
</tr>
<tr>
<td>VntxClearFD1</td>
<td>VntxClearFD</td>
<td>SESS=&quot;last&quot;</td>
</tr>
<tr>
<td>VntxClearDoc1</td>
<td>VntxClearDoc</td>
<td>SESS=&quot;last&quot;</td>
</tr>
</tbody>
</table>
The XML template is loaded by `VntxLoadLast`, and elements are populated using the `VntxSetTag` function to tell Siebel eBusiness, along with various identifying information, which account numbers to use (elements `FundsTransferFromAccountNumber` and `FundsTransferToAccountNumber`), and how much to transfer (element `FundsTransferDollarAmount`). The returned document is then stored, an element is read, and form data is cleared.
Appendix A   References

Siebel eBusiness


CONVERSANT

We’d like your opinion.

Avaya welcomes your feedback on this information product. Your comments can be of great value in helping us improve the information that supports our products.

CONVERSANT System Version 8.0 Application Development with Siebel eBusiness, 585-310-784, Issue 1, June 2001

1. Please rate the effectiveness of this document in the following areas:

<table>
<thead>
<tr>
<th></th>
<th>Excellent</th>
<th>Good</th>
<th>Fair</th>
<th>Poor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ease of Finding Information</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clarity</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Completeness</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accuracy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Organization</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Appearance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Examples</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Illustrations</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overall Satisfaction</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. Please check the ways you feel we could improve this document:

- Improve the overview/introduction
- Improve the table of contents
- Improve the organization
- Add more figures
- Add more examples
- Add more detail
- Make it more concise
- Add more step-by-step procedures/tutorials
- Add more troubleshooting information
- Make it less technical
- Add more/better quick reference aids
- Improve the index

Please add details about your concern.

_________________________________________________________________________________
_________________________________________________________________________________

3. What did you like most about this document?

_________________________________________________________________________________
_________________________________________________________________________________

4. Feel free to write any comments below or on an attached sheet.

_________________________________________________________________________________
_________________________________________________________________________________

If we may contact you concerning your comments, please complete the following:

Name: ____________________________________________ Telephone Number: ( )
Company/Organization __________________________ Date: _________________
Address: __________________________

When you have completed this form, please fax to (303) 538-1741. Thank you.