



CONVERSANT[®] System
Version 8.0
Application Development with
Siebel eBusiness

585-310-784
Issue 2
September 2001

Notice

Every effort was made to ensure that the information in this book was complete and accurate at the time of printing. However, information is subject to change.

Preventing Toll Fraud

"Toll fraud" is the unauthorized use of your telecommunications system by an unauthorized party (for example, a person who is not a corporate employee, agent, subcontractor, or working on your company's behalf). Be aware that there may be a risk of toll fraud associated with your system and that, if toll fraud occurs, it can result in substantial additional charges for your telecommunications services.

Avaya Fraud Intervention

If you *suspect that you are being victimized* by toll fraud and you need technical support or assistance, call Technical Service Center Toll Fraud Intervention Hotline at +1 800 643 2353.

Providing Telecommunications Security

Telecommunications security (of voice, data, and/or video communications) is the prevention of any type of intrusion to (that is, either unauthorized or malicious access to or use of your company's telecommunications equipment) by some party.

Your company's "telecommunications equipment" includes both this Avaya product and any other voice/data/video equipment that could be accessed via this Avaya product (that is, "networked equipment").

An "outside party" is anyone who is not a corporate employee, agent, subcontractor, or working on your company's behalf. Whereas, a "malicious party" is anyone (including someone who may be otherwise authorized) who accesses your telecommunications equipment with either malicious or mischievous intent.

Such intrusions may be either to/through synchronous (time-multiplexed and/or circuit-based) or asynchronous (character-, message-, or packet-based) equipment or interfaces for reasons of:

- Utilization (of capabilities special to the accessed equipment)
- Theft (such as, of intellectual property, financial assets, or toll-facility access)
- Eavesdropping (privacy invasions to humans)
- Mischief (troubling, but apparently innocuous, tampering)
- Harm (such as harmful tampering, data loss or alteration, regardless of motive or intent)

Be aware that there may be a risk of unauthorized intrusions associated with your system and/or its networked equipment. Also realize that, if such an intrusion should occur, it could result in a variety of losses to your company (including but not limited to, human/data privacy, intellectual property, material assets, financial resources, labor costs, and/or legal costs).

Your Responsibility for Your Company's Telecommunications Security

The final responsibility for securing both this system and its networked equipment rests with you - an Avaya customer's system administrator, your telecommunications peers, and your managers. Base the fulfillment of your responsibility on acquired knowledge and resources from a variety of sources including but not limited to:

- Installation documents
- System administration documents
- Security documents
- Hardware-/software-based security tools
- Shared information between you and your peers
- Telecommunications security experts

To prevent intrusions to your telecommunications equipment, you and your peers should carefully program and configure your:

- Avaya-provided telecommunications systems and their interfaces
- Avaya-provided software applications, as well as their underlying hardware/software platforms and interfaces
- Any other equipment networked to your Avaya products.

Federal Communications Commission Statement

Part 15: Class A Statement. This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at his own expense.

Part 15: Class B Statement. This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses, and can radiate radio-frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient the receiving television or radio antenna where this may be done safely.
- To the extent possible, relocate the receiver with respect to the telephone equipment.
- Where the telephone equipment requires ac power, plug the telephone into a different ac outlet so that the telephone equipment and receiver are on different branch circuits.

Part 15: Personal Computer Statement. This equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC Rules. Only peripherals (computing input/output devices, terminals, printers, etc.) certified to comply with the Class B limits may be attached to this computer. Operation with noncertified peripherals is likely to result in interference to radio and television reception.

Part 68: Network Registration Number. This equipment is registered with the FCC in accordance with Part 68 of the FCC Rules. It is identified by FCC registration number xxx.

Part 68: Answer-Supervision Signaling. Allowing this equipment to be operated in a manner that does not provide proper answer-supervision signaling is in violation of Part 68 rules. This equipment returns answer-supervision signals to the public switched network when:

- Answered by the called station
- Answered by the attendant
- Routed to a recorded announcement that can be administered by the CPE user

This equipment returns answer-supervision signals on all DID calls forwarded back to the public switched telephone network. Permissible exceptions are:

- A call is unanswered
- A busy tone is received
- A reorder tone is received

Canadian Department of Communications (DOC) Interference Information

This digital apparatus does not exceed the Class A limits for radio noise emissions set out in the radio interference regulations of the Canadian Department of Communications.

Le Présent Appareil Numérique n'émet pas de bruits radioélectriques dépassant les limites applicables aux appareils numériques de la class A prescrites dans le règlement sur le brouillage radioélectrique édicté par le ministère des Communications du Canada.

European Union Declaration of Conformity

Avaya Business Communications Systems declares that equipment specified in this document conforms to the referenced European Union (EU) Directives and Harmonized Standards listed below:

EMC Directive 89/336/EEC

Low Voltage Directive 73/23/EEC



The "CE" mark affixed to the equipment means that it conforms to the above Directives.

Trademarks

-- *CONVERSANT* is a registered trademark of Avaya, Inc.
-- *Vonetix* is a registered trademark of Gold Systems, Inc.
-- *Siebel* is a registered trademark of Siebel Systems, Inc.

Ordering Information

Call: Avaya Publications Center
Voice +1 800 457 1235
Fax +1 800 457 1764
International Voice +1 410 568 3680
International Fax +1 410 891 0207

Write: Globalware Solutions
200 Ward Hill Avenue
Haverhill, MA 01835 USA
Attention: Avaya Account Manager

Email: totalware@gwsmail.com

Order: Document No. 585-310-784, Issue 2
September 2001

You can be placed on a Standing Order list for this and other documents you may need. Standing Order will enable you to automatically receive updated versions of individual documents or document sets, billed to account information that you provide. For more information on Standing Orders, or to be put on a list to receive future issues of this document, please contact the Avaya Publications Center.

Warranty

Avaya Inc. provides a limited warranty on this product. Refer to the "Limited use Software License Agreement" card provided with your package.

Avaya National Customer Care Center

Avaya provides a telephone number for you to use to report problems or to ask questions about your contact center. The support telephone number is 1-800-242-2121.

Avaya Web Page

<http://www.avaya.com>

Comments

To comment on this document, return the comment card at the end of the document.

Acknowledgment

This document was written by the CRM Development group of Avaya University

CONVERSANT System
Version 8.0
Application Development with Siebel eBusiness

Contents

Introduction

Prerequisites	vii
Basic concepts	viii
Creating an application	ix

Identifying data to be transferred

Data on Siebel eBusiness	1
Data on CONVERSANT	2

Setting up Siebel

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
Voice@Work application detail	22
Passing calls and information to a Siebel agent	23
Balance queries and fund transfers.	26

Query 27

Setting up a transfer 31

Updating Siebel eBusiness to perform a transfer 35

Appendix A: References

Siebel eBusiness 39

CONVERSANT 39

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:

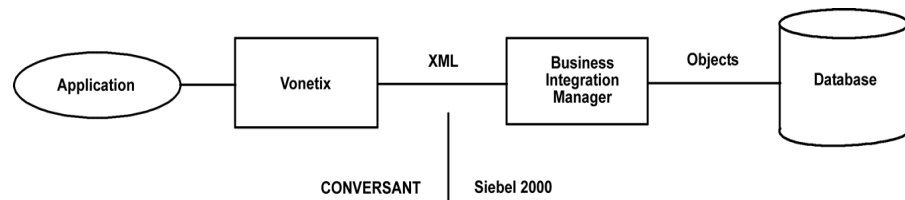
- Familiarity with Siebel eBusiness (setup, administration, and use).
- Familiarity with CONVERSANT application development, preferably using Voice@Work.
- Familiarity with the Extensible Markup Language (XML).

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 below, 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.

If the Computer Telephony Integration (CTI) feature is being used, then a call can be passed to the Siebel operator from CONVERSANT. Information inserted by the application into the UI field will then automatically be transferred to the Siebel eBusiness Desktop (as a “screen pop”). For information regarding the CTI feature, see *CONVERSANT System Version 8.0 Computer Telephony Integration*, 585-352-200.

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: Setting up Siebel on page 3.
4. Create a workflow within Siebel eBusiness that manages the exchange of information between Siebel eBusiness and CONVERSANT. See Chapter 2: Setting up Siebel on page 3.
5. Use Siebel eBusiness to generate a document type definition (DTD) for each integration object. See Chapter 2: Setting up Siebel 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 Setting up Siebel

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.

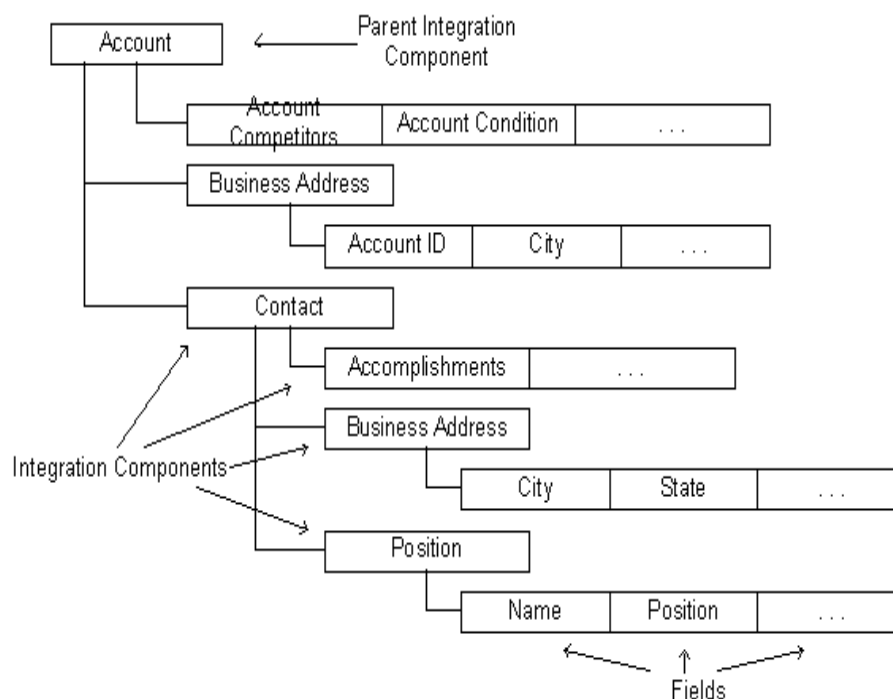
See the *Siebel 2000 Bookshelf* for more information.

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

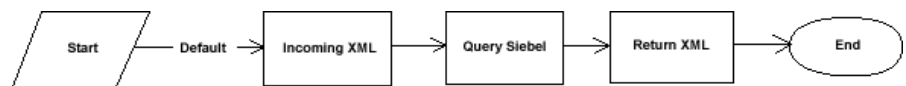


Workflows

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

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

```
<!-- 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

```
<!-- 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,

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

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

Slightly more complicated is the line

```
<!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

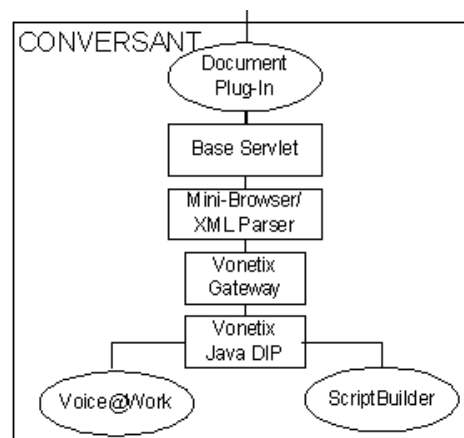
- Prepare and send outgoing XML documents.
- Receive and process incoming XML documents.

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.

CONVERSANT and the Vonetix Interface



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 `vntxLoadDoc (SESSIONNAME, FILENAME)` function, where the argument `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 `vntxSetTag (SESSIONNAME, TAGNAME, TAGNUMBER, TAGVALUE)` function for each element identified by the arguments `TAGNAME` and `TAGNUMBER`. The content of the element is substituted for `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 `vntxSetFD (SESSIONNAME, FIELDNAME, FIELDVALUE)` function, where the argument `FIELDNAME` is the field's name, and the argument `FIELDVALUE` is the field's value.

Each of the following fields needs to be defined:

- a. **SWEExtSource**

The workflow name (Business Service Source), found in the Siebel eBusiness config file.

- b. **SWEExtCmd**

The command to be executed by Siebel eBusiness. The most commonly used command is likely to be **Execute**.

- c. **UserName**

The login name.

- d. **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

VntxSetDocFD(*SESSIONNAME*, *name*) function does this, where *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 **vntxPostDoc**(*SESSIONNAME*, *URL*) function, where the argument *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 **vntxStoreDoc**(*SESSIONNAME*, *FILENAME*) function, where the argument *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 **vntxGetTag**(*SESSIONNAME*, *TAGNAME*, *TAGNUMBER*, *TAGTEXT*) function. *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 **vntxClearSes**(*SESSIONNAME*) function to clear *both* the XML document and HTTP request form data.
- Use the **vntxClearDoc**(*SESSIONNAME*) function to clear just the form data encoded XML document.
- Use the **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 to edit. One option is to speak to an operator.
4. The caller is prompted for the account number.
5. If the caller chose to speak to an operator, then Step 6 will be followed. If the caller chose any other option, then Step 8 will be followed.
6. The application uses the CTI feature to send the account number to Siebel eBusiness via the CVCT server. The account number then appears as a screen pop on the Siebel desktop. CTI is used by the application to transfer the call to a Siebel agent.
7. The call ends.
8. 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.
9. The second menu prompts the caller for the type of account to transfer money from and the type of account to transfer money to.
10. The caller is prompted to enter the account numbers for the two account types chosen in Step 9.
11. The caller is prompted to enter the amount of money to be transferred.
12. Siebel eBusiness is queried for the validity of the account numbers

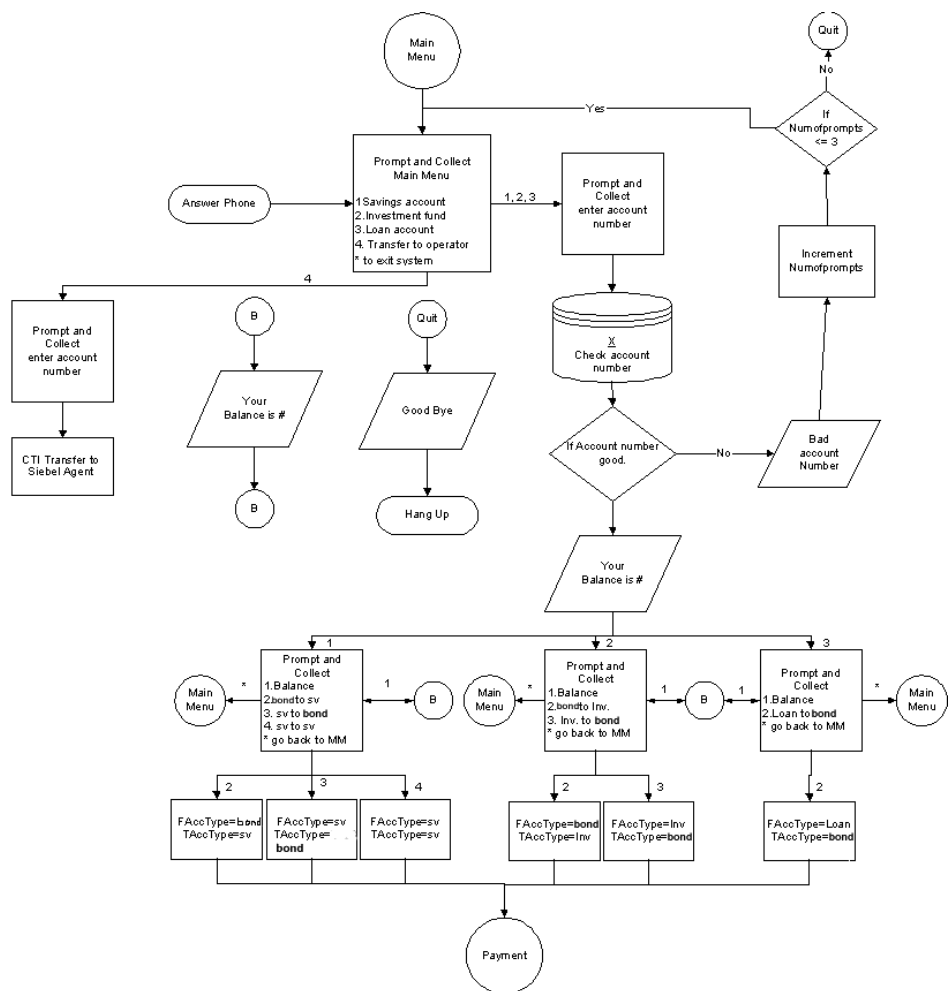
Any errors are reported to the caller, who is then routed back to the main menu.

- If the transfer is not confirmed, then the caller is routed to the main menu.

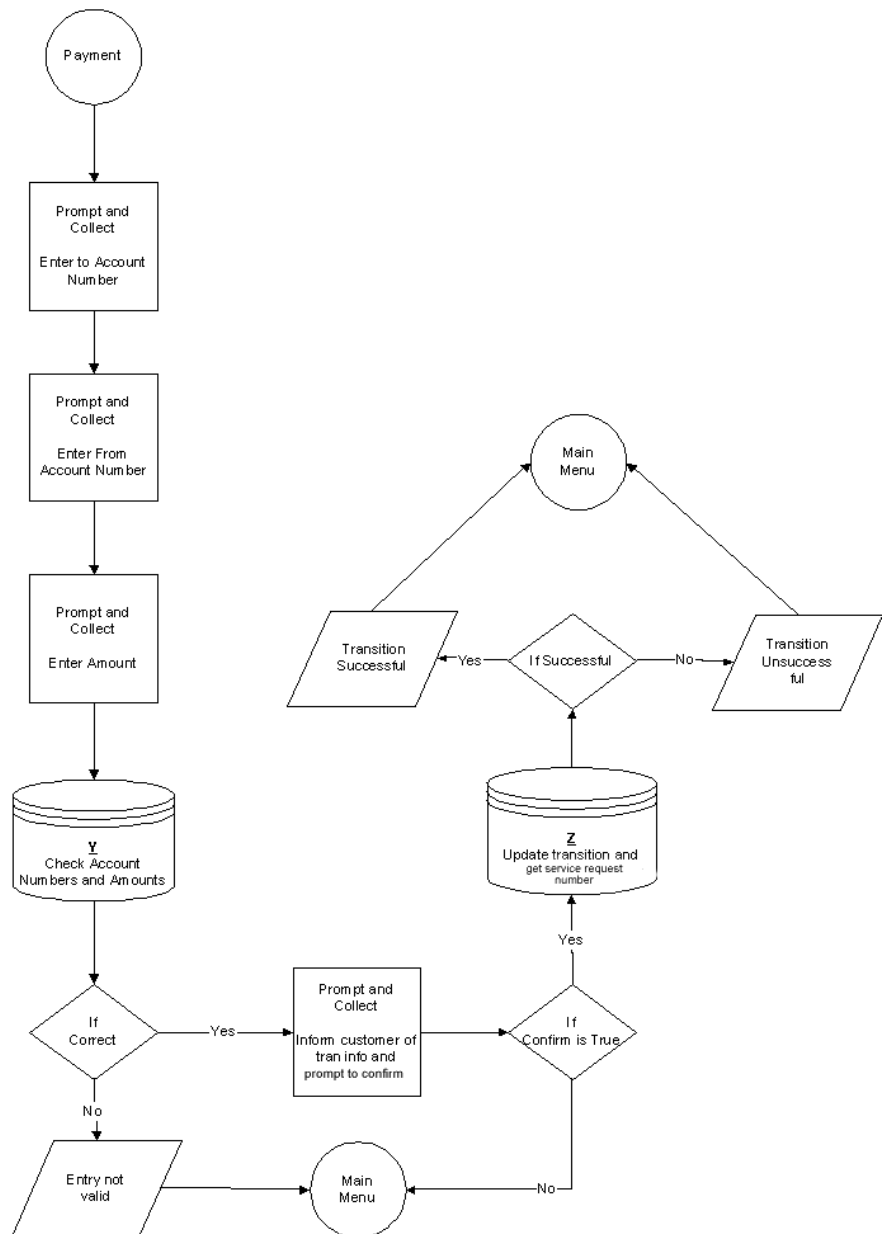
- If there is an error, then the caller is informed of the error and then routed to the main menu.

16. The caller is routed to the main menu.

Sample application flowchart - 1



Sample application flowchart - 2



XML documents

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

Query

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

Query returned

```
<?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

```
<SiebelMessage MessageId=""
MessageType="Integration Object"
IntObjectName="Bubba Funds Transfer SR">
<BubbaFundsTransferSr>
<FundsTransferSR>
<ContactFinancialAccounts>999-999-
999</ContactFinancialAccounts>
<ContactLastName>Jordan</ContactLastName>
<FundsTransfer-FromAccountNumber>111-111-
111</FundsTransfer-FromAccountNumber>
<FundsTransfer-ToAccountNumber>555-555-
555</FundsTransfer-ToAccountNumber>
<FundsTransfer-DollarAmount>20000</FundsTransfer-
DollarAmount>
<TransactionId>IVR-Transfer-3</TransactionId>
</FundsTransferSR>
</BubbaFundsTransferSr>
</SiebelMessage>
```

Update returned

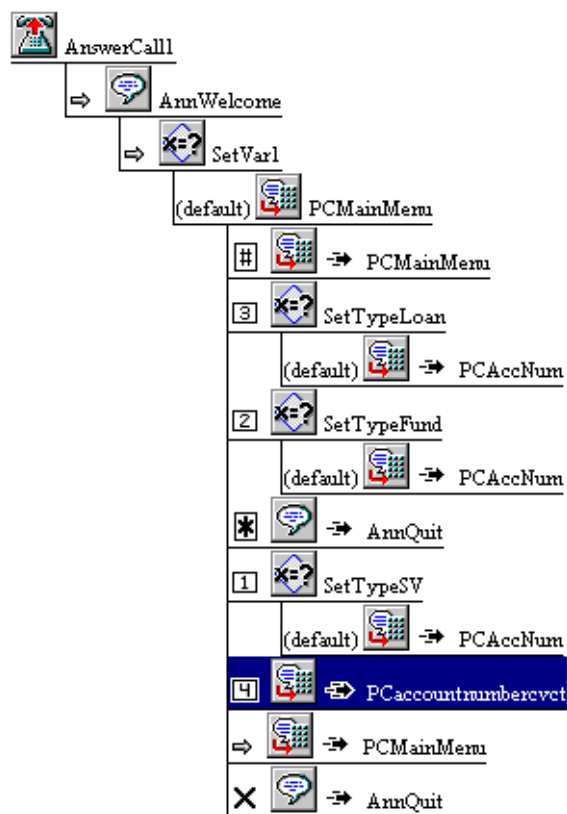
```
<?xml version="1.0" encoding="UTF-8"?>
<SiebelMessage IntObjectName="Bubba Funds Transfer
SR" MessageId="" MessageType="Integration Object">
<BubbaFundsTransferSr>
<FundsTransferSR>
<ContactFinancialAccounts>999-999-
999</ContactFinancialAccounts>
<ContactLastName>Jordan</ContactLastName>
<FundsTransferFromAccountNumber>111-111-
111</FundsTransferFromAccountNumber>
<FundsTransferToAccountNumber>555-555-
555</FundsTransferToAccountNumber>
<FundsTransferDollarAmount>20000</FundsTransferDol
larAmount>
<TransactionId>IVR-Transfer-3</TransactionId>
<SRNumber>1-NAX</SRNumber> </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 ("Main menu" 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 to select an account to edit. One of the options is to speak to an operator.

Main menu



Passing calls and information to a Siebel agent

If the caller chooses to talk to the operator, the application jumps to `PCaccountnumbercvct`.

At the `PCaccountnumbercvct` node, the caller is prompted for an account number. The account number is stored as a variable.

The following actions, shown in "PCaccountnumbercvct detail" on page 25 and described below, are function calls to the CTI DIP.

NOTE:

For information about the use of the CTI DIP, see *CONVERSANT System Version 8.0 Computer Telephony Integration*, 585-352-200.

The call ID, ANI, DNIS, station extension, and skill/split hunt group number are collected with the `ctiCallInfo` function (at `CtiCallInfo1`), and the caller is placed on hold (with the `ctiHold` function at node `CtiHold1`).

The application calls the Siebel agent with the `ctiDial` function at node `CtiDial1`. The `ctiDial` function also passes the account number as UUI information to Siebel eBusiness.

The `ctiNotify` function at `CtiNotify1` lets the application know whether the Siebel agent answers this call. The call ID of the call transferred to the Siebel agent is also collected at this point.

The caller and Siebel agent can talk to each other when the `ctiTransfer` function is invoked by the application (at node `CtiTransfer1`).

NOTE:

The call ID of both the caller and the Siebel agent need to be known so they can be used as arguments for the `ctiTransfer` function. The call ID of the caller was collected at `CtiCallInfo1`, and the call ID of the Siebel agent was collected at `CtiNotify1`.

At node `CtiCallState2`, the application uses the function `ctiCallState` to determine if the call is still present on the CONVERSANT port.

If the call is present, then the application frees up the CONVERSANT port, leaving the caller and Siebel agent to talk independently of CONVERSANT. The application does this by:

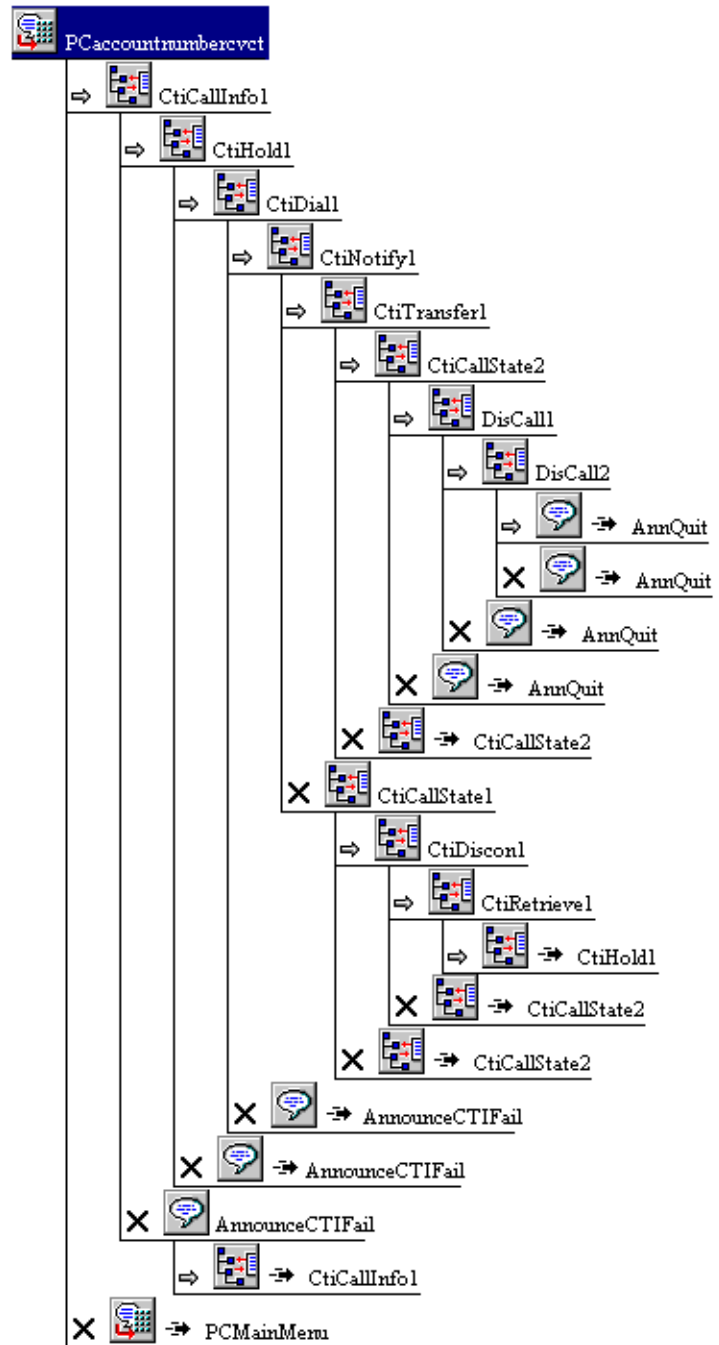
1. removing the connection with the caller with a `ctiDiscon` function call at node `DisCall1`

2. removing the connection with the Siebel agent with a **ctiDiscon** function call at node **DisCall2**

The remaining actions deal with failures. There are just a few new nodes introduced here.

- **AnnQuit** is an announcement thanking the caller for calling.
- **CtiRetrieve1** uses the **ctiRetrieve** function to take the caller out of the hold state. This makes the held call the active call.
- **AnnounceCTIFail** is an announcement notifying the caller that all agents are busy.

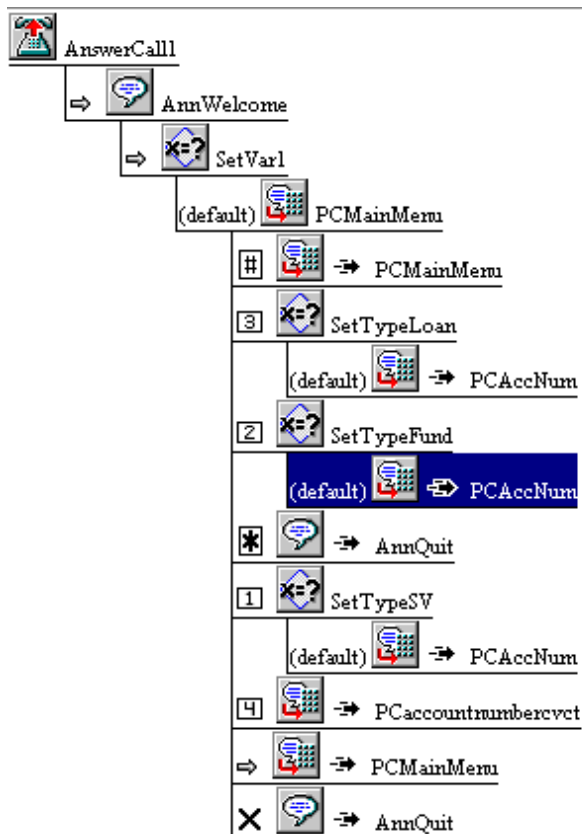
PCaccountnumbercvct detail



Balance queries and fund transfers

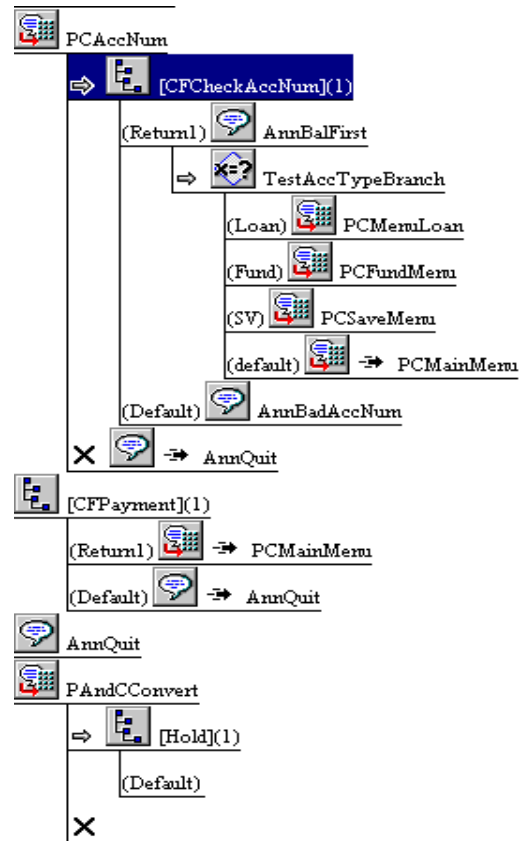
If the caller chooses not to talk to an operator at the main menu, then the application will perform a balance query and fund transfer. The main menu is shown again in the figure below.

Main menu



PCAccNum (see "PCAccNum detail" on page 27) sets up and executes both a balance query and the transfer of funds from one account to another.

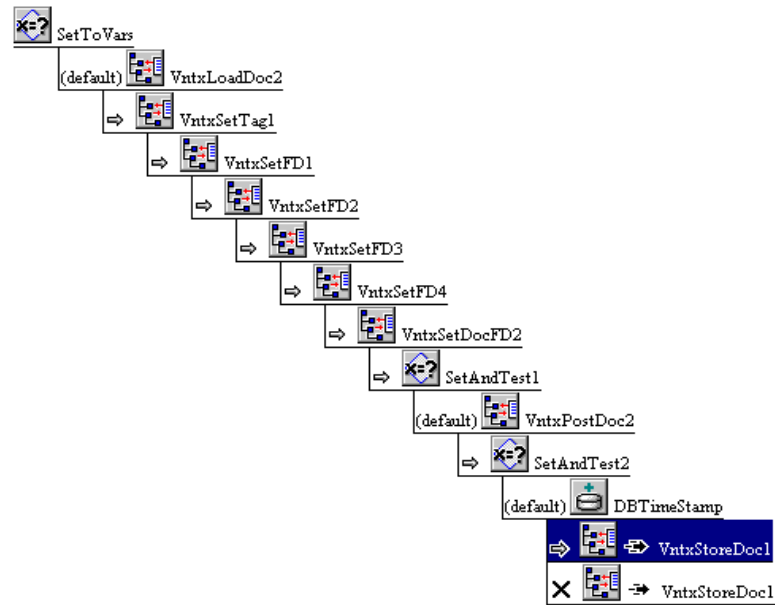
PCAccNum detail



Query

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 28.

CFCheckAccNum detail



Node	Function	Arguments
VntxLoadDoc2	VntxLoadDoc	SESS="firstacc", FILE="/home/bankdemo/accQuery.xml"
VntxSetTag1	VntxSetTag	SNAME="firstacc", TNAME="AccountNumber", TNUMB=1, VALUE=[AccNum]
VntxSetFD1	VntxSetFD	SESS="firstacc", NAME="SWEEExtSource", TEXT="xmlquery"
VntxSetFD2	VntxSetFD	SESS="firstacc", NAME="SWEEExtCmd", TEXT="Execute"
VntxSetFD3	VntxSetFD	SESS="firstacc", NAME="UserName", TEXT="SADMIN"
VntxSetFD4	VntxSetFD	SESS="firstacc", NAME="Password", TEXT="SADMIN"

Node	Function	Arguments
VntxSetDocFD2	VntxSetDocFD	SESS="firstacc", DOC="SWEExtData"
VntxPostDoc2	VntxPostDoc	SESS="firstacc", URL="http://.../eai/start.swe"

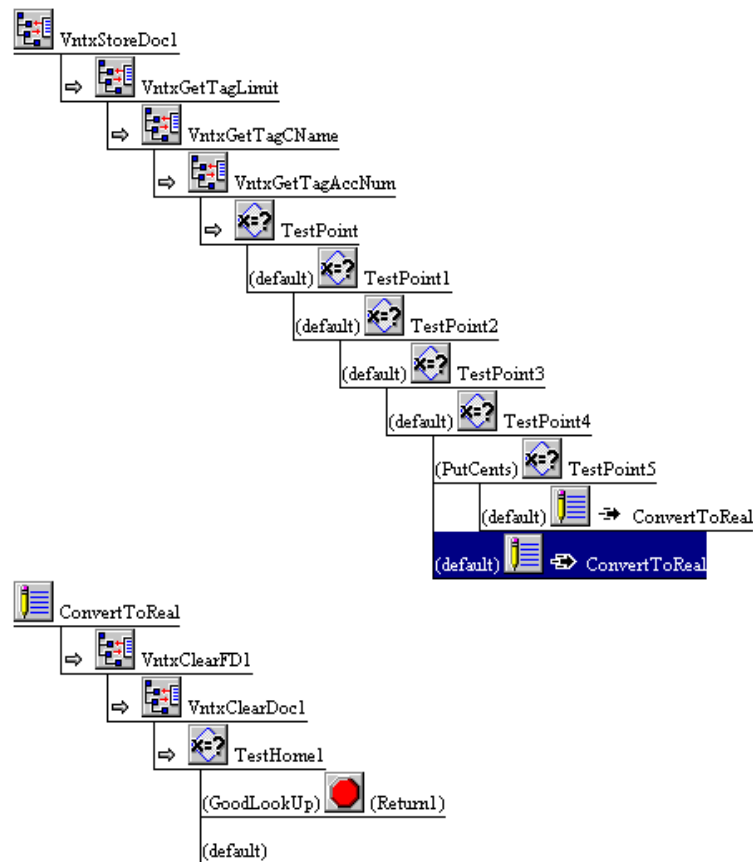
The XML template is loaded (**VntxLoadDoc2**), and is used as the outgoing XML document after having its **AccountNumber** element populated (in **VntxSetTag1**) with the account number that Siebel eBusiness will use to look up the information wanted.

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 30.

VntxStoreDoc1 detail



Node	Function	Arguments
VntxStoreDoc1	VntxStoreDoc	SNAME="firstacc", FNAME="/home/bankdemo/out/a ccmaster.out"
VntxGetTagLimit	VntxGetTag	SNAME="firstacc", TNAME="RelationShipLimit", TNUMB=1, ATEXT=[MainAccLimit]
VntxGetTagCName	VntxGetTag	SNAME="firstacc", TNAME="LastName", TNUMB=1, ATEXT=[ContactLastName]

Node	Function	Arguments
VntxGetTagAccNum	VntxGetTag	SNAME="firstacc", TNAME="CurrentBalance", TNUMB=1, ATEXT=[CurrBalance]
VntxClearFD1	VntxClearFD	SESS="firstacc"
VntxClearDoc1	VntxClearDoc	SESS="firstacc"

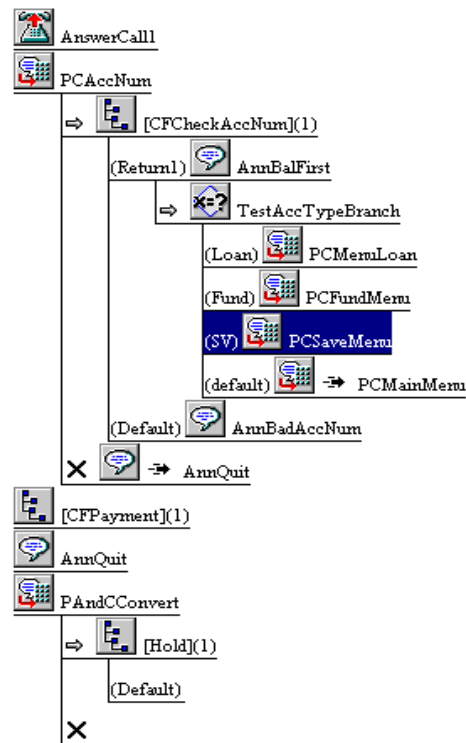
VntxStoreDoc1 uses the **VntxStoreDoc** function to store the XML document sent from Siebel eBusiness. Elements of interest are then read from the XML document using the **VntxGetTag** function (this function name is part of the relevant node names in the figure above). After some manipulation of the 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**).

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 32). 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:

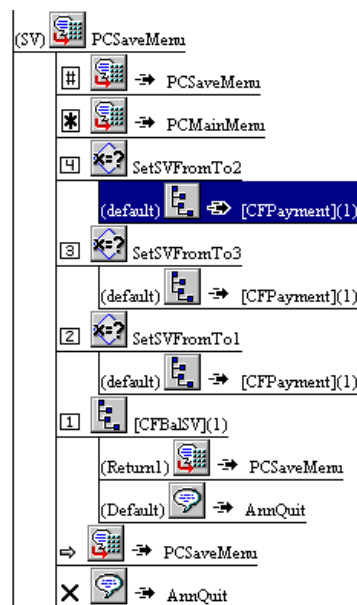
- **PCMenuLoan** is entered if the account type was "Loan".
- **PCFundMenu** is entered if the account type was "Fund".
- **PCSaveMenu** is entered if the account type was "Save".

Second menu detail



PCSaveMenu is typical of the process used for prompting the caller for accounts and for transferring funds, as shown in "PCSaveMenu detail (from PCAccNum)" on page 32:

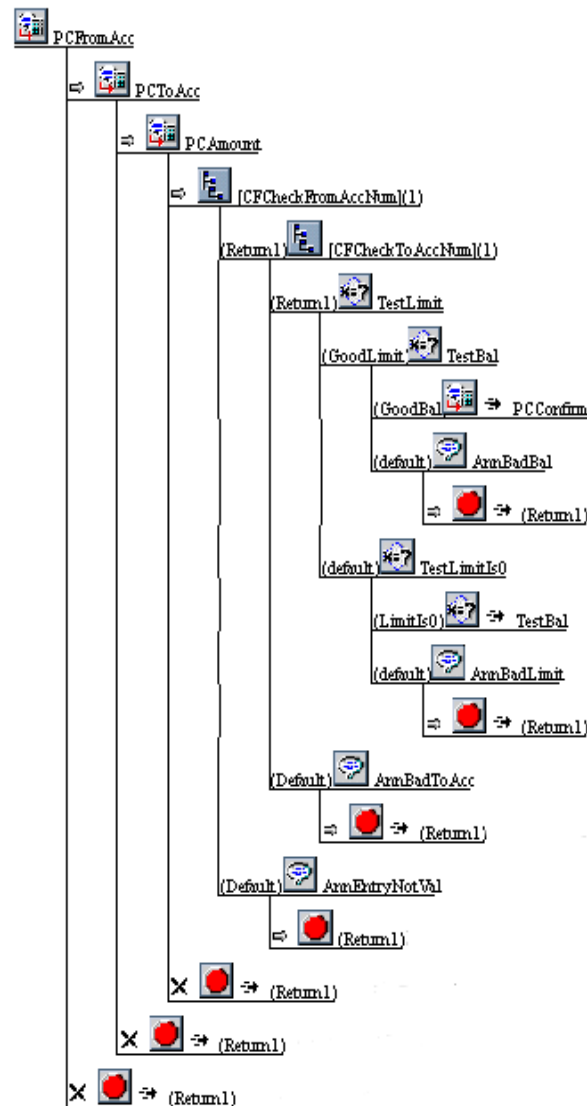
PCSaveMenu detail (from PCAccNum)



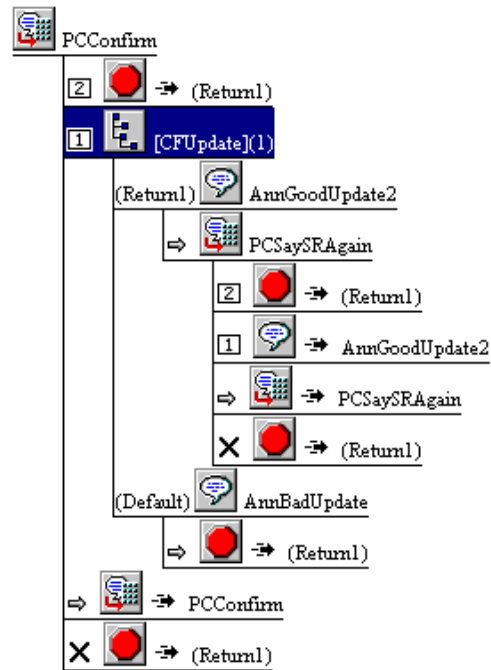
The caller selects which account money will be transferred from, and which account the money will be transferred to. The application then branches to the appropriate call flow (**SetSVFromTo2**, **SetSVFromTo3**, or **SetSVFromTo1**) and performs the transfer (**CFPayment**).

CFPayment starts with two account number queries (see "CFPayment detail" on page 34). **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 28).

CFPayment detail

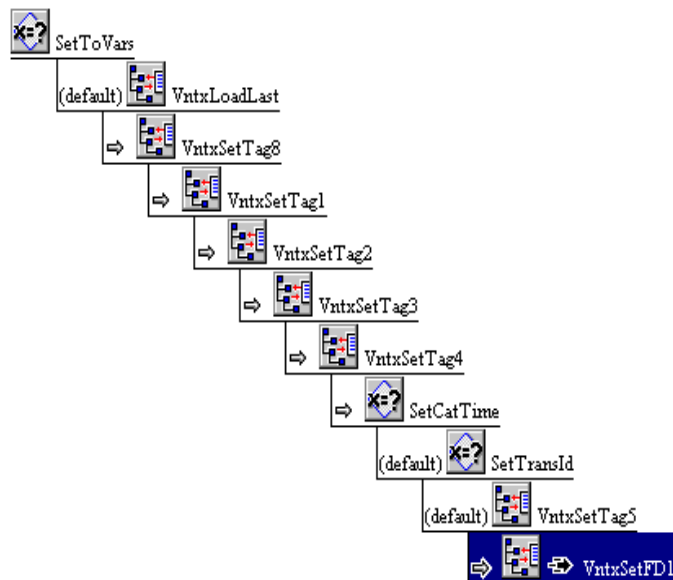


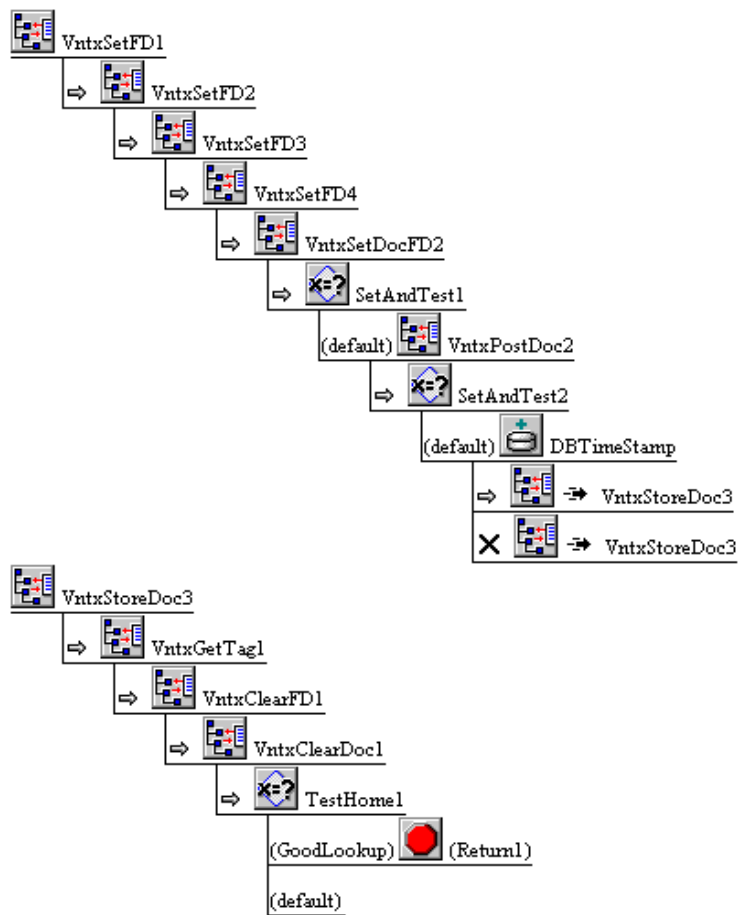
The actual transfer of money, in the form of a Siebel eBusiness update, occurs in **PCCConfirm** (see "PCCConfirm detail" on page 35).

PCConfirm detail

Updating Siebel eBusiness to perform a transfer

CFUpdate has the update steps, as shown in "CFUpdate detail" on page 35.

CFUpdate detail



Node	Function	Argument
VntxLoadLast	VntxLoadDoc	SESS="last", FILE="/home/bankdemo/accUpdate.xml"
VntxSetTag8	VntxSetTag	SNAME="last", TNAME="ContactFinancialAccounts", TNUMB=1, VALUE=[AccNum]
VntxSetTag1	VntxSetTag	SNAME="last", TNAME="ContactLastName", TNUMB=1, VALUE=[ContactLastName]

Node	Function	Argument
VntxSetTag2	VntxSetTag	SNAME="last", TNAME="FundsTransferFromAc countNumber", TNUMB=1, VALUE=[FAccNum]
VntxSetTag3	VntxSetTag	SNAME="last", TNAME="FundsTransferToAcco untNumber", TNUMB=1, VALUE=[TAccNum]
VntxSetTag4	VntxSetTag	SNAME="last", TNAME="FundsTransferDollarA mount", TNUMB=1, VALUE=[TransAmount]
VntxSetTag5	VntxSetTag	SNAME="last", TNAME="TransactionID", TNUMB=1, VALUE=[TransID]
VntxSetFD1	VntxSetFD	SESS="last", NAME="SWEExtSource", TEXT="xmlupdate"
VntxSetFD2	VntxSetFD	SESS="last", NAME="SWEExtCmd", TEXT="Execute"
VntxSetFD3	VntxSetFD	SESS="last", NAME="UserName", TEXT="SADMIN"
VntxSetFD4	VntxSetFD	SESS="last", NAME="Password", TEXT="SADMIN"
VntxSetDocFD2	VntxSetDocFD	SESS="last", DOC="SWEExtData"
VntxPostDoc2	VntxPostDoc	SESS="last", URL="http://.../eai/start.swe"
VntxStoreDoc3	VntxStoreDoc	SNAME="last", FNAME="/home/bankdemo/out/a ccUpdate.out"
VntxGetTag1	VntxGetTag	SNAME="last", TNAME="SRNumber", TNUMB=1, ATEXT=[SRNum]

Node	Function	Argument
VntxClearFD1	VntxClearFD	SESS="last"
VntxClearDoc1	VntxClearDoc	SESS="last"

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

- *Siebel 2000 Bookshelf* (Siebel Systems, 2000).
-

CONVERSANT

- *CONVERSANT System Version 8.0 Computer Telephony Integration*, 585-352-200.
- *Vonetix 2.1 Developer Guide* (Gold Systems, 2000).

For more information, see <http://www.goldsys.com/products/>.

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 2, September 2001

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

	Excellent	Good	Fair	Poor
Ease of Finding Information				
Clarity				
Completeness				
Accuracy				
Organization				
Appearance				
Examples				
Illustrations				
Overall Satisfaction				

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

- | | |
|--|---|
| <input type="checkbox"/> Improve the overview/introduction | <input type="checkbox"/> Make it more concise |
| <input type="checkbox"/> Improve the table of contents | <input type="checkbox"/> Add more step-by-step procedures/tutorials |
| <input type="checkbox"/> Improve the organization | <input type="checkbox"/> Add more troubleshooting information |
| <input type="checkbox"/> Add more figures | <input type="checkbox"/> Make it less technical |
| <input type="checkbox"/> Add more examples | <input type="checkbox"/> Add more/better quick reference aids |
| <input type="checkbox"/> Add more detail | <input type="checkbox"/> 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.

