

# ICAS for eCust

Moving Customer Access Into The  
Future



## ICAS client platform requirement (New GUI)

- >32 MB RAM, >120 MHz CPU
- VGA Display
- Window 95, 98, NT (both Work Station and Server)
- Internet Explorer 4.0.1 or later
- Microsoft JVM (Java Virtual Machine) build 3165 or later

## Web Server

- Any

## IGATE

- OrbixWeb 2.0.1 (soon will be upgrade to 3.1c)
- Platform: any platform supported by OrbixWeb
- IGATE server from CyberObject

Note: IGATE can be installed on the same machine as the Web server or can be installed on a different machine on a different location from the Web server.

## Client Software Installation

- No ICAS client installation required
- New version will be placed in the Web server and be automatically distributed/installed to the client machine.
- Software is downloaded only once unless new version is available on the Web server.

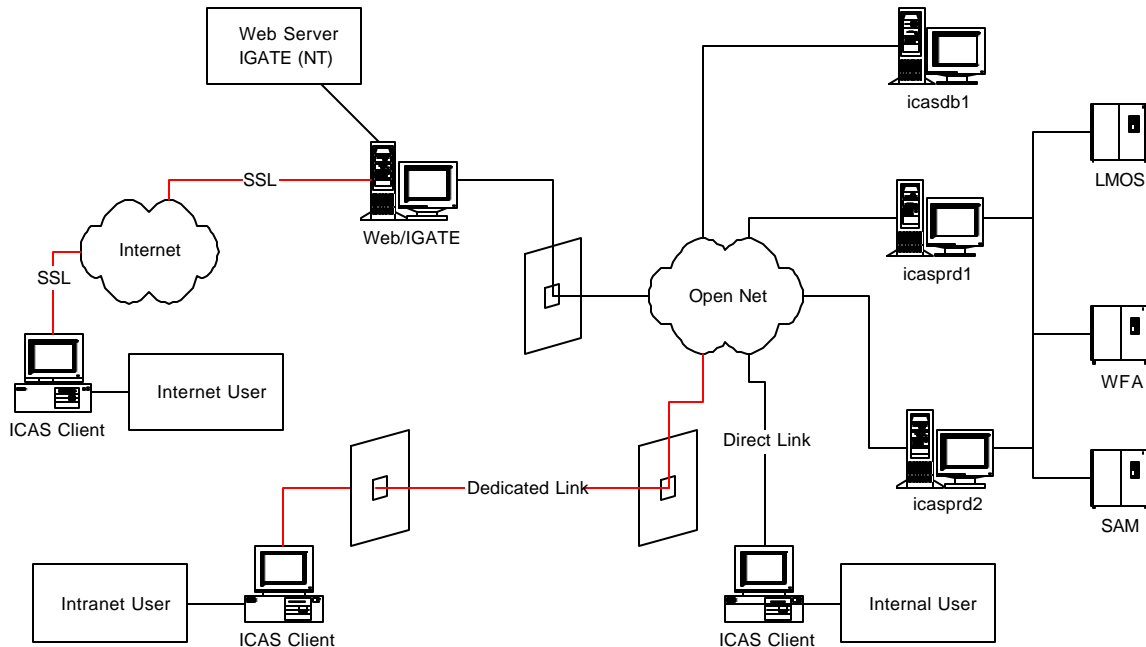
## Security

- Digital certificate via HTTP/Web
- SSL session between client and IGATE
- Firewall support
- Access authorization via ICAS server
- Record protection via ICAS server
- Extensive access log by ICAS server

## ICAS Deployment Diagram

A new element is introduced to accommodate the Internet user: IGATE.

The Internet user cannot access the ICAS server directly. IGATE (ICAS GATEway) is introduced to bridge the two party. Major function of IGATE is routing traffic between the Internet ICAS client and the ICAS server.



Once the ICAS active object embedded in a web page is activated on the ICAS client machine, the ICAS active object will establish a SSL session back to the IGATE. With a SSL session, it really means:

- It is only ICAS client with ICAS application certificate that shall be able to establish a connection to the IGATE server (host and port). The ICAS application certificates are distributed by CA (Certificate Authority) at Ameritech. The CA host (a machine for managing application certificate such as generating, deleting certificate, etc.) can be standalone without connecting it to any network. The CA host needs to be very secure so that nobody will be able to steal certificate.
- IGATE will authenticate the ICAS client via certificate.
- ICAS client will authenticate IGATE via certificate.
- Data traffic between ICAS client and IGATE is encrypted.

If someone broke into the IGATE host (NT), he would not be able to submit any request to the back-end systems via IGATE without significant efforts for the following reasons:

- He has to write an ICAS client to send any request via IGATE.

- He will need CORBA stubs code to compile the client and IGATE host does not contain that.
- He has to know the object and method signatures to make any CORBA calls.

Because the ICAS client and IGATE use a SSL session to communicate, so nobody will be able to decipher the above information by sniffing the TCP/IP traffic between them.

It would be multiple times harder for someone to break the IGATE than to figure out how to run a CGI program on the same host.

## An Overview of SSL Security

SSL provides authentication, privacy, and integrity for communications across TCP/IP connections. Authentication allows an application to verify the identity of another application with which it communicates. Privacy ensures that data transmitted between applications cannot be eavesdropped on or understood by a third party. Integrity allows applications to detect if data was modified during transmission.

## Authentication in SSL

SSL uses Rivest Shamir Adleman (RSA) public key cryptography for authentication. In public key cryptography, each application has an associated public key and private key. Data encrypted with the public key can be decrypted only with the private key. Data encrypted with the private key can be decrypted only with the public key.

Public key cryptography allows an application to prove its identity by encoding data with its private key. As no other application has access to this key, the encoded data must derive from the true application. Any application can check the content of the encoded data by decoding it with the application's public key.

## The SSL Handshake Protocol

Consider the example of two applications, a client and a server. The client connects to the server and wishes to send some confidential data. Before sending application data, the client must ensure that it is connected to the required server and not to an impostor.

When the client connects to the server, it confirms the server identity using the SSL handshake protocol. A simplified explanation of how the client executes this handshake in order to authenticate the server is as follows:

- The client initiates the SSL handshake by sending the initial SSL handshake message to the server.
- The server responds by sending its certificate to the client. This certificate verifies the server's identity and contains its public key.
- The client extracts the public key from the certificate and encrypts a symmetric encryption algorithm session key with the extracted public key.
- The server uses its private key to decrypt the encrypted session key that it will use to encrypt and decrypt application data passing to and from the client. The client will also use the shared session key to encrypt and decrypt messages passing to and from the server. For a complete description of the SSL handshake, refer to the Netscape Communications SSL V3.0 specification, available from [www.netscape.com](http://www.netscape.com). The SSL protocol permits a special optimized handshake in which a previously established session can be resumed. This has the advantage of not needing expensive public key computations. The

SSL handshake also facilitates the negotiation of ciphers to be used in a connection. The SSL protocol also allows the server to authenticate the client.

As any application can have a public and private key pair, the transfer of the public key must be accompanied by additional information that proves the key is associated with the true server and not some other application. For this reason, the key is transmitted as part of a certificate.

## Certificates in SSL Authentication

The public key is transmitted as part of a certificate. A certificate is used to ensure that the public key submitted is in fact the public key that belongs to the submitter. For the certificate to be acceptable to the client, it must have been digitally signed by a Certificate Authority (CA) that the client explicitly trusts.

The International Telecommunications Union (ITU) recommendation X.509 defines a standard format for certificates. SSL authentication uses X.509 certificates to transfer information about an application's public key.

An X.509 certificate includes the following data:

- The name of the entity identified by the certificate.
- The public key of the entity.
- The name of the certification authority that issued the certificate.

The role of a certificate is to match an entity name to a public key. A CA is a trusted authority that verifies the validity of the combination of entity name and public key in a certificate.

According to the SSL protocol, it is unnecessary for applications to have access to all certificates. Generally, each application only needs to access its own certificate and the corresponding issuing certificates. Clients and servers supply their certificates to applications that they want to contact during the SSL handshake. The nature of the SSL handshake is such that there is nothing insecure in receiving the certificate from an as yet untrusted peer. The certificate will be checked to make sure that a trusted CA has digitally signed it and the peer will have to prove its identity during the handshake.

## Privacy of SSL Communications

When a client authenticates a server, confidential data sent by the client can be encoded by the server's public key. It is only the actual server application that will be able to decode this data, using the corresponding private key.

Immediately after authentication, an SSL client application sends an encoded data value to the server. This unique session encoded value is a key to a symmetric cryptographic algorithm. A symmetric cryptographic algorithm is an algorithm in which a single key is used to encode and decode data. Once the server has received such a key from the client, all subsequent communications between the applications can be encoded using the agreed symmetric cryptographic algorithm. This feature strengthens SSL security.

Examples of symmetric cryptographic algorithms used to maintain privacy in SSL communications are the Data Encryption Standard (DES) and RC4.

## **Integrity of SSL Communications**

The authentication and privacy features of SSL ensure that applications can exchange confidential data that cannot be understood by an intermediary. However, these features do not protect against the modification of encrypted messages transmitted between applications.

To detect if an application has received data modified by an intermediary, SSL adds a message authentication code (MAC) to each message. This code is computed by applying a function to the message content and the secret key used in the symmetric cryptographic algorithm. An intermediary cannot compute the MAC for a message without knowing the secret key used to encrypt it. If the message is corrupted or modified during transmission, the message content will not match the MAC. SSL automatically detects this error and rejects corrupted messages.

## **Contact Information:**

### **Sales Marketing**

Christopher Reed

Tel: (404) 842-0015

Fax: (404) 264-1154

### **Corporate Offices**

CyberObject Corporation

3050 A-1 Business Park Drive

Norcross, GA 30071

Tel: (678) 969-1515

Fax: (678) 969-1514

Web: <http://www.cyberobject.com>

