Pennsylvania

Department of Human

Services

Bureau of Information Systems

DHS OpenTI Component Standards

Version 2.1

December 27th, 2004

Table of Contents

OpenTI and OLTP Terminology

Introduction

The establishment of a common terminology is imperative for understanding new technologies. Effective communication requires that the meaning of technical terminology is agreed upon by all parties. To that end this document begins with the definitions of terms used in OpenTI and OLTP. These definitions are to be interpreted within the context of these technologies.

Terms

Transaction Manager

A transaction manager is a run-time environment that provides various services to applications, including transactional services.

ACID

ACID is an acronym that defines the characteristics of a “transaction”. A transaction is Atomic, Consistent, Isolated and Durable. To support ACID, a Transaction Manager may coordinate its activities with Resource Managers (e.g. database managers). A Transaction Manager may also coordinate its activities with other Transaction Managers.

Two Phase Commit

Also referred to as 2PC. Two Phase Commit is a process whereby two (or more) transaction managers manage the state of an application transaction to support the ACID properties. The transaction managers collaborate to ensure that the entire transaction either succeeds (Commit) or fails (Rollback) in an ACID fashion. The Transaction Managers interact with their respective Resource Managers.

COM+

COM+ is Transaction Manager for the Windows operating system. A feature of COM+ known as XA provides an interface that can be used to collaborate with transaction managers that support the X/Open protocol, such as OLTP.

OLTP

OLTP (On Line Transaction Processing) is a Transaction Manager that runs on the Unisys OS2200 platform. It uses the X/Open protocol to collaborate with other transaction managers that are X/Open-compatible.

OpenTI

OpenTI (Open Transaction Integrator) is middleware for the Windows operating system that enables the COM+ Transaction Manager to work with Transaction Managers that use the X/Open protocol. It accomplishes this using the XA feature of COM+

Client Server

Client Server is a paradigm where two parties operate in a request / response fashion, with each taking on the role of a requestor (client) or a responder (server).

OLTP Server

A Unisys mainframe program may be packaged so as to be available as a service thru OLTP. So packaged, the program is part of an OLTP Server. It responds to requests from clients, such as OLTP Clients or OpenTI Clients.

OpenTI Server

OpenTI provides the means to build a COM+ compatible interface (Server Definition) that can be called from OLTP via OpenTI. This interface may be implemented by a COM+ application program. This COM+ program may then be linked to the Server Definition via OpenTI. Thus the COM+ application becomes an OpenTI Server. It responds to requests from OLTP Clients.

OpenTI Client

OpenTI provides the means to build a COM+ compatible component that that can call an OLTP server via OpenTI. This component may be instantiated on the Windows operating system to send requests to that OLTP Server.

OLTP Client

A Unisys mainframe program may make calls to OLTP services that enable it to call OLTP Servers or OpenTI Servers. Such a program is an OLTP Client.

Windows-Initiated OpenTI

Also known as Web-Initiated OpenTI. At DHS we often use the term Windows-Initiated OpenTI to refer to a scenario where an OpenTI Client is communicating with an OLTP Server. This term is not technically robust, but it is valid in the current environment at DHS.

Mainframe-Initiated OpenTI

Also known as Host-Initiated OpenTI. At DHS we often use the term Mainframe-Initiated OpenTI to refer to a scenario where an OLTP Client is communicating with an OpenTI Server. This term is not technically robust, but it is valid in the current environment at DHS.

OpenTI / OLTP Admin Guidelines

Introduction

The purpose of this section is to describe the procedures for developing and maintaining applications that use UNISYS' OLTP and OpenTI on Windows® NT/2000.

Definition

OpenTI is a UNISYS product that puts a Microsoft COM wrapper around an OLTP service. Conversely, it can expose COM+ services so they can be used by the Unisys mainframe. The services must conform to the X/Open standard. OpenTI may take advantage of the Intel side of a UNISYS Clearpath IX host.

Development Process

Developing a system that participates in the component-based environment generally follows the steps outlined below. These steps are similar for both mainframe-initiated and windows-initiated transactions.

1. Abstract application processes to maximize reusability: In a 2-tiered environment, application processes tend to be monolithic. In an n-tiered component based environment, it is best to breakdown business processes to as fine a level of granularity as possible. This maximizes the degree of reusability making for rapid solution development and enables the ability to rapidly respond to new challenges. The mainframe and Microsoft developers working together best perform this process.

2. Define the inputs and outputs: This process is no different in a component-based environment than in any other. It is important to finalize this definition early to avoid any undo and/or redo scenarios further down the line. The mainframe and Microsoft developers working together best perform this process. The middleware components should act solely as pipelines, transforming the data from the source environment format tot the target environment format. No business logic should be implemented here.

3. Create the Views: A V*iew* is an X/Open compliant buffer format used to move data into and out of X/Open transactions. Each view accurately describes the inputs and/or outputs defined in step 2 above. The symbolic format of a view is very simple but its content must be consistent for all entities that will use it. Views are generally specified by the mainframe programmer and created by the OLTP administrator. Views are at the core of the interface. As such they are under the control of a single administration point.

4. Code the Solution: Step 1-3 above requires the participation of both the mainframe and Microsoft developers. Once these steps are complete, the process may split into parallel efforts as follows:

    a. Mainframe Developers: Code the OLTP Services.

    b. OpenTI Administrators: Create the OpenTI components.

    c. Microsoft Component Developers:

        a) Create the OpenTI Wrapper Component.

        b) Create the Business Component using the Wrapper.

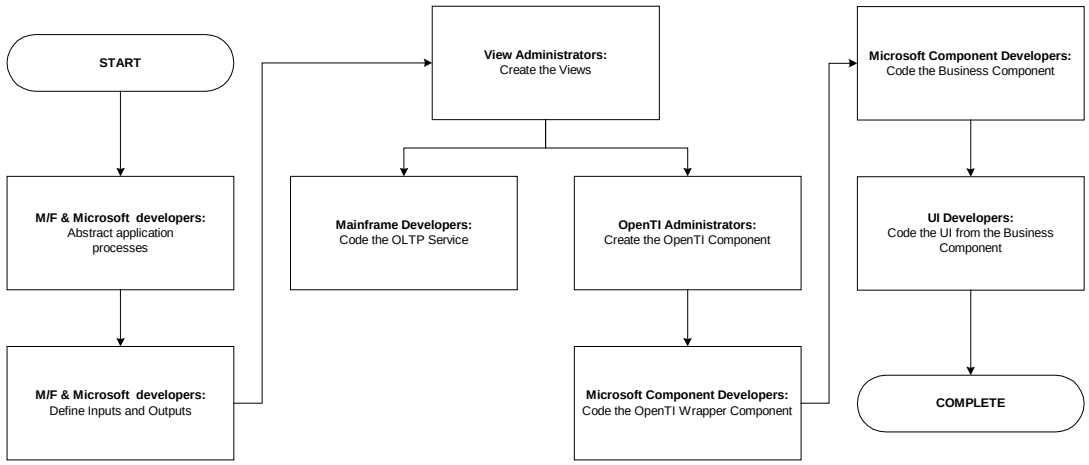
        c) Create SOAP Wrapper for Business Component.

    d. User Interface (UI) Developers:

        a) Microsoft developers may create UI from VB, PowerBuilder, ASP, or any COM+ compliant development environment using the Business Component.

b) Java or other non-Microsoft developers may use Business Components via SOAP or .Net wrappers for UI development.

This process is outlined in the diagram below:



A more complete process flow diagram can be found in the section “Consolidated Flow”.

Environments

Development is implemented on a variety of servers. The OpenTI development system is a commodity desktop system. IKE-C is registered as the OLTP host for this system. Its purpose is to provide a developer's environment that includes the standard suite of support and development software, including the OpenTI development environment. The environment code is "**U**".

Administration

Views

The symbolic form of a view is a simple text file. Any text editor on the host (e.g. @ED, @IPF) or Windows (e.g. Notepad, Word) could be used to create the views. However, since their content must be consistent for all entities that will use them, their creation and maintenance must be carefully administered.

Administration

DHS has authorized a limited number of individuals to create and maintain Views. Developers are required to forward a request form via email to one of these individuals for view creation. The format of the request form is similar to the existing OLTP Server and OLTP Service request forms (see Attachment A). The request includes a description of the fields (name, size, type) to be passed. The developer will provide all specifications for the view

Symbolic Format

The first line of a view identifies the view name (case-sensitive). The lines that follow detail the name and format of the fields in the view. Comment lines are allowed - prefixed with an ampersand ('#'). A view defines the fields in the buffer. An example of a view that defines individual fields follows. Note that this view does not follow current DHS practice. It is shown to demonstrate the capability of a view to define the individual fields in the data buffer.

```
VIEW V0003AURIU
# TYPE      CNAME             FBNAME  COUNT     FLAG      SIZE    NULL
# ----      -----             ------  -----     ----      ----    ----
```

```
  char      TRANSACTION_CODE       -      6        -         -      -

  char      RECIPIENT_NUMBER       -     10        -         -      -

END
```

Current DHS practice is to define each view as a single character field of the required length. The fields in the view are defined on the OLTP side by a proc. The fields in the view must be parsed in code on the Windows side. As a consequence, only ASCII characters may be used in view buffers. Views are defined by their direction, length and application area. Additional space is commonly provided in the view. These practices drastically reduce the effort required to implement changes to these interfaces.

Notice also that Views are not environment-specific. They are applicable for the entire OLTP runtime environment in which they are registered.

Following view is an example of a view that follows current DHS practice. It defines an output buffer of 400 characters and is used in the fictitious AUR application area.

```
VIEW VAUR00004000

# TYPE   CNAME                    FBNAME    COUNT          FLAG    SIZE
       NULL

# ------ ------------------------ ------    -------        ----    -------
       ----

  CHAR   VAUR00004000      -          400            -      -              -

END
```

The version of OpenTI currently used at DHS has a limit of 16k per field in views used in OpenTI Clients. If more than 16k of data must be represented in the buffer, it can be represented in multiple fields of appropriate length. The following view is an example of a view that uses this technique to define a buffer of 84000 characters.

```
VIEW VAUR00840000

# type   cname                          fbname count   flag size    null
```

```
# ------ ------------------------------- ------ ------- ---- ------- ----

#

  CHAR   VAUR00160010                     -      16000  -    -       -

  CHAR   VAUR00160020                     -      16000  -    -       -

  CHAR   VAUR00160030                     -      16000  -    -       -

  CHAR   VAUR00160040                     -      16000  -    -       -

  CHAR   VAUR00160050                     -      16000  -    -       -

  CHAR   VAUR00040000                     -       4000  -    -       -

END
```

Symbolic Creation

Views are created by developers on the Unisys mainframe platform. They are sent to the View administrator for cataloging in OLTP. They are then sent to Windows platform via FTP and emailed to the developers on the Windows/Intel platform. Copies of the views are used as needed on the Windows/Intel platform.

Maintenance of the Views is done on the Unisys mainframe platform.

The Unisys mainframe programmer defines a proc that defines the views fields. A “DCD” listing of that proc is helpful to the Windows programmer who must write code to parse the data in the view buffer.

Access

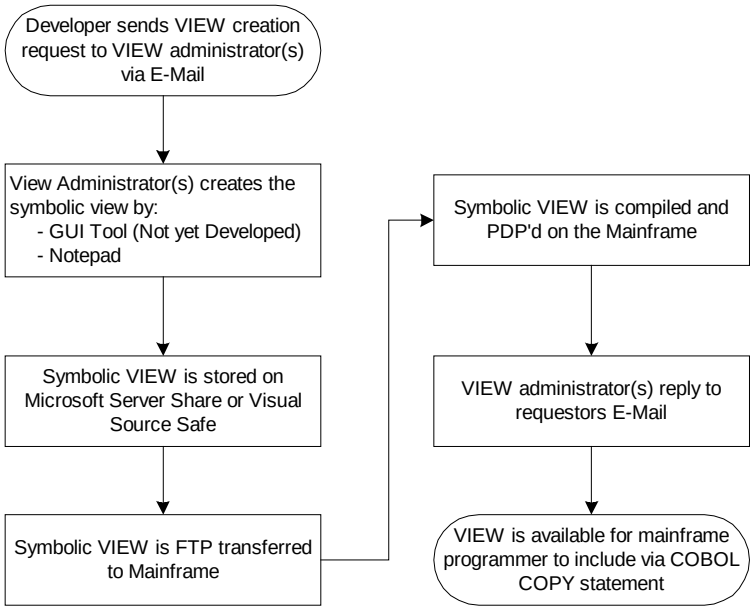
On the Microsoft node, read only access is granted on views to all authorized developers. Write access is restricted the view administrator(s) and protected in VSS.

On the mainframe, the program files to which the views are FTP transferred, are write-protected by Security Option 1 Access Control Records (ACRs). Only the view administrator(s) know the write key.

Compilation

Once a symbolic view is stored on the host, it must be compiled and PDP'd to be available for inclusion in a COBOL program as a COPY Proc. The compilation is accomplished with a specialized processor called @VADD. This process is either triggered by the FTP transfer process or is done as a separate process by the view administrator(s). Once the view is available, the view administrator(s) informs the requesting development personnel by replying to the initial email.

Flow (see section Consolidated Flow for overview)



OpenTI Components

An OpenTI component serves as a Microsoft COM+ wrapper around an OLTP service that exists on the mainframe.

Description

Windows-Initiated OpenTI

Microsoft COM+ provides a remote procedure calling technology that can be thought of as synonymous with 2200 common banks. However, each COM object may have multiple *entry points*. In component terminology, each *entry point* is referred to as a method. Each method call can pass a collection of parameters. OpenTI transitions the parameters on the Microsoft COM call to/from OLTP view buffers that are then forwarded to the appropriate service on the mainframe.

Typically, each OpenTI component is associated with a single OLTP service and is created from two view buffers: One for input and the other for output. At runtime, the client program making the request calls a component's method with a list of parameters. The parameters are paired sequentially to each field in the input and output views. The OpenTI runtime transitions the input parameters to the input view, calls the OLTP service, receives an output view, transitions the output view to the output parameters and returns control to the calling client.

Mainframe-Initiated OpenTI

In the other direction, Mainframe-Initiated OpenTI is similar, but the roles are reversed. It uses the capability of COM+ for an application to implement an interface that is defined elsewhere. A Mainframe-Initiated call originates in an OLTP transaction and is received by OpenTI. OpenTI refers to the registration information for this service to determine the COM+ component to call and what interface to use when calling it. The interface is defined by the OpenTI Server Definition that was built using the service name and the views associated with the transaction. The COM+ component is called, executes its processing, and returns data to OpenTI via the output view.

OpenTI consists of 2 products:

a) OpenTI Runtime - described above

b) OpenTI Builder - a GUI program used to create the components

A copy of the OpenTI Builder is installed on the designated OpenTI development machine.

Administration

A development server is maintained by BTE for the explicit use of the development teams to do server application development, including OpenTI development. Developers build the OpenTI components and OpenTI Server Definitions on the development server using the Views that were developed by the View administrator. Component configuration and unit testing is done on the development server.

When unit testing is completed, the developers created the OpenTI components for the remaining environments and install them on the next level servers. Then each additional level of testing is performed. These levels are Integration Testing and Systems Acceptance Testing.

When all testing phases are successfully completed, the developers build the OpenTI components for the Production environment and submit the appropriate Move Requests to Quality Assurance. The developers may be asked to participate in the implementation of the components on the production servers.

Naming

Please reference the “Service Layer Component” section of this standard for a more detailed description of COM+ naming formats. (See “OpenTi Service Component Naming” section within the "OLTP/OpenTI Naming Standard".)

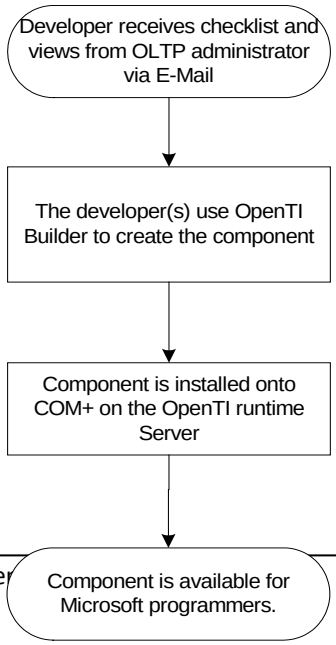
Component Creation

OpenTI Builder is a simple point and click GUI program.

For Windows-Initiated OpenTI, the OpenTI Builder is used to build the OpenTI Component. It prompts the user for the location of the views (normally, one for input and another for output) and the name of the components. The views are retrieved from the server share directory or VSS database where they are stored and maintained by the view administrator(s). Once created, the components are placed into COM+ on the server where the OpenTI runtime is installed. The component is then available for use.

For Mainframe-Initiated OpenTI, the OpenTI Builder is used to build the OpenTI Server Definition. It prompts the user for the location of the views (normally, one for input and another for output) and the name of the service. The views are retrieved from the server share directory or VSS database where they are stored and maintained by the view administrator(s). Once created, the server definition is registered on the server. The server definition is referenced in the worker component in VB via the “Implements”. Finally the OpenTI Management Console is used to correlate the OpenTI Server Definition and the implementation component. The component is then available for use.

Flow (see section Consolidated Flow for overview)



OLTP Services and Servers

Each OLTP service can be thought of as synonymous with a TIP transaction. However, the service is not attached to or dependent on the origin of the input. Significantly, they do not originate from or respond with a screen. This is why OLTP services fit so nicely into a component based environment.

Services are logically and functionally collected into an OLTP Server. Each service is coded and functions as a subroutine. The services are collected into a Server with the Make-A-Server (MAS) processor.

It is these OLTP Servers and Services that can be exposed to COM+ via OpenTI, thus forming the foundation of Windows-Initiated OpenTI.

Administration

Servers and Services are configured on the mainframe in a symbolic file and processed by the TMSCON processor in a fashion similar to VALTAB administration. The compiled and mapped absolutes or zooms must be loaded into an appropriate program library also in similar fashion to TIP transactions.

DHS has designated the Enterprise Operating Systems Section within the Division of Systems Engineering, Bureau of Technology and Engineering to serve as the OLTP administrator group. There are three entities on the mainframe to administer: The Servers, the Services, and the program libraries. Program libraries will be administered by the Quality Assurance Unit.

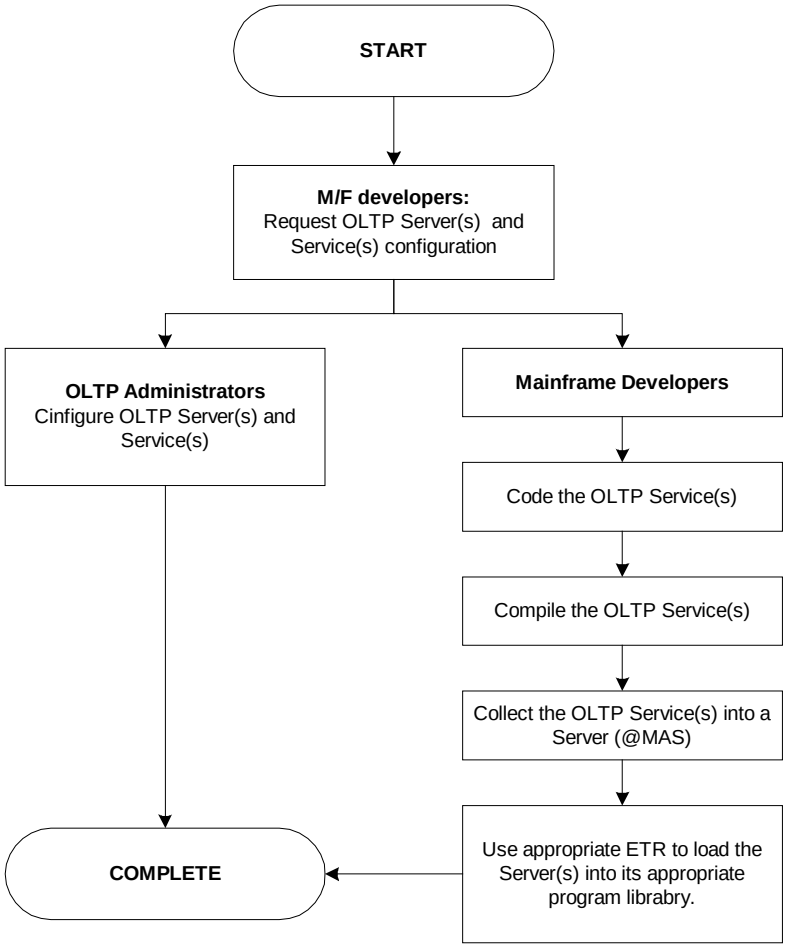
1. Servers - A request form for server configuration is used (see Attachment A).
2. Services - A request form for server configuration is used (see Attachment A).

3. Program Libraries - DHS currently has a process called Element Transfer Process (ETR) for moving programs into production. This process is used for TIP and batch programs. This process has been enhanced to accommodate OLTP programs as well.

Naming

DHS created a naming convention that is described within this document. For more information on view, server service, and remote service names, see the section "OLTP/OpenTI Naming Standard".

Flow (see section Consolidated Flow for overview)



OpenTI Server Definitions

An OpenTI Server Definition is a COM dll that defines the interface that OpenTI will use to call an implementation component. It can be thought of as a proxy in the sense that it does not incorporate any application logic. Rather, its sole responsibility is to refer to the component that does do the work, the implementation component.

OpenTI Server Definitions are registered on the target server. They are also referenced in the implementation component using the “Implements” keyword. Finally, they are used in the OpenTI Management Console to correlate the OpenTI Server Definition to the Implementation Component.

Administration

Server Definitions are built by the Windows developers using the OpenTI Builder. Deployment is done by the developers on platforms where they have access. On other platforms deployment is done by those with proper privileges under the guidance of the Windows developers.

As with other Windows components, the OpenTI Server Definitions and Implementation components are housed in Visual Source Safe. Production deployments are administered by the Quality Assurance Unit and implemented by the Bureau of Infrastructure Management and Operation.

Naming

DHS created a naming convention within this document. For more information on view, server service, and server definition, see the section "OLTP/OpenTI Naming Standard".

OpenTI/OLTP Naming Standard On the Unisys 2200

This section describes the mainframe naming standards in force at the Pennsylvania Department of Welfare for the OpenTI and OLTP technologies.

Introduction

The Department of Human Services has standardized on products such as Open OLTP, Open Transaction Integrator, and OSI-TP. UNISYS created the OpenTI proprietary middleware to connect Legacy Data architectures to the N-Tier architecture via COM+. The use of these products required DHS to develop naming conventions for use in its enterprise.

Name Case

All Names used for VIEWS, SERVERS, and SERVICES are Case Sensitive and must be in UPPER CASE for both the OS2200 and NT, W2K, and Enterprise server platforms. Note that the COM+ Business Objects, Service Wrappers and OpenTI Components may use mixed case, however the Service Name within the OpenTI Components must match the 2200 Clearpath specification. (See "Service Layer Components" chapter of this document.)

View Names

When creating view names for Data Access Services (DAS) used by Unisys OpenTI and Open OLTP this naming standard is applied. Each View representing data in an OLTP or Open/TI service must take on these properties with no delimiters “-“, “_”, “/”, or “,” applied to it’s name. **/VIEW** is the standard element suffix format generated for COBOL elements.

Naming Convention for Views

Name Format: **VAAANNNNNNN{O|I}**

Element Format: **VAAANNNNNNN{O|I}/VIEW**

COBOL Proc Name: **VAAANNNNNNN{O|I}**

Explanation:

V (char. 1): Indicates the object is a View.

AAA (char. 2-4): Application-Id a three (3)-character acronym for the system/subsystem. It may consist of upper case letters and/or numbers. The Application-Id must begin with a letter.

NNNNNNN (char. 5-11): A number from 1 to 9999999. Indicates the length of the view.

O|I (char. 9): An upper case alpha character that indicates the type of view. There are presently two (2); however, there can be up to 26.

O = Output view.

I = Input view.

/VIEW (char. 11-15): The mass storage element must have a version name of VIEW.

Examples:

1. Create an AUR input view.

   View Name: VAUR0001234I

   Element Name: VAUR0001234I/VIEW

   COBOL Proc Name: VAUR0001234I

   View: #V0001AMSI Legacy replacement for DPS screen num: 154 (see note 4)

   Cobol: *V0001AMSI Legacy replacement for DPS screen num: 154 (see note 4)

2. Create an AUR output view.

   View Name: VAUR0004123O

   Element Name: VAUR0004123O/VIEW

   COBOL Proc Name: VAUR0004123O

Notes:

1. The view name and the mass storage element name are to be the same. Additionally, the mass storage element is to have a version name of VIEW.

2. All internal/external/documentation references to a view name must be consistent with this naming standard.

3. By having the view name begin with “V”, it will be easily distinguishable from other elements in a typical programmers work file.
   In this standard Views are treated as generic objects. Views are named by their application area, length, and mode (Input or Output). Views are not associated by name to their associated components or services. Views are used only define a buffer of a given length, not to define the layout of data being transported within them.

4. Where views are created to replace legacy TIP/DPS screens, they are to be documented both within the view and the COBOL program as follows:

   View:

   #VAAANNNNNNN{I|O} OLTP/OpenTI replacement for Legacy {TIP|DPS} screen num: NNNN

   COBOL:

   *VAAANNNNNNN{I|O} OLTP/OpenTI replacement for Legacy {TIP|DPS} screen num: NNNN

5. Note that the View Name does not include characters that indicate the environment where the view is used. A single view is always used for all environments. This reduces the amount of code that is changed when moving an OpenTI process from one environment to the next.

Server Names

The following discussions on Server Name and Service Name apply to the development of OLTP Servers. See the Glossary for the definition of an OLTP Server and how they relate to other clients.

The Server Name in Unisys Open Transaction Integrator (Open/TI) reflects the configuration name used to identify a set of Services that communicate from COM+ to the Unisys 2200.

Unisys reserves 12 characters to describe the server name. However, if you are going to convert the server to run in a BATCH, TIP or HVTIP environment, the SERVER NAME restriction is reduced to 6 characters. Within the Identification Division of a COBOL program, the Program-id's first six characters must match the first six characters of the SERVER NAME followed by SVR. This server name will then be entered into the OS configuration file for OLTP servers using forms in Attachment A.

Service Names

The SERVICE NAME defines a unique identifier up to 32 characters long (only the first 15 characters are significant). The SERVICE NAME is the name used by the client to request work from a specific service routine. One or more Services may be implemented in a single Server.

The 1st six characters of the SERVICE NAME must mach the SERVER NAME. The Program-id within the Identification Division of a COBOL program must match the first 6 characters of the SERVICE NAME followed by SVC and the Sequence Number

Remote Service Name (RNAME)

The Remote Service Name defines the name of a service on a remote system that will be called by an OLTP Client. The Remote Service Name generally refers to an OpenTI Server Definition. It is a unique identifier up to 32 characters long (only the first 15 characters are significant). The Remote Service Name is specified as the RNAME parameter where the OLTP Client is defined in the OLTP Config. It does not require a VALTAB registration.

Naming Convention for OLTP Servers

Name Format: **ENNAAASVR**

Element Format: **ENNAAASVR**

Explanation:

E (char.1): The environment within which the Server operates. There are six (6) standard environments;
U = Unit test,
I = Integration test,
T = Training,
S = Systems Acceptance Test,
F = Test For Production,
P = Production

NN (char. 2-3): Sequence number from 1 to 99. It indicates the server number. There can be up to 99 servers in a particular system/subsystem. Always start with 01 for the 1st Server within an applications area of service.

AAA (char. 4-6): Application-Id a three (3)-character acronym for the system/subsystem. It may consist of upper case letters and/or numbers. The Application-Id must begin with a letter.

SVR (char. 7-9): Indicates OLTP/OpenTI Server Transaction:

Examples:

1. Create the first server for the AUR production environment:

   Server & Element Name: P01AURSVR

2. Create two (2) AUR servers, one providing host services and the second for remote services. Both operate in the unit test environment.

   Server 1: U01AURSVR Server 2: U02AURSVR

Notes:

1. The first twelve (12) characters of the server name and the mass storage element name are to be the same.

2. All internal/external/documentation references to a service name must be consistent with this naming standard.

3. Notice that the Server name includes a character that indicates the environment for that service. This is necessary in order to support the multiple environments that are supported in a single instance of TIP on the Unisys 2200 system.

Naming Convention for OLTP Services

Name Format: **ENNAAASVCNN**

Element Format: **ENNAAASVCNN**

Explanation:

E (char.1): The environment within which the Server operates. There are six (6) standard environments;
U = Unit test,
I = Integration test,
T = Training,
S = Systems Acceptance Test,
F = Test For Production,
P = Production

NN (char. 2-3): A number from 1 to 99. Indicates the server number. There can be up to 99 servers in a particular system/subsystem.

AAA (char. 4-6): A three (3) character acronym for the system/subsystem. It may consist of upper case letters and numbers. It must, however, begin with a letter.

SVC (char. 7-9): Indicates that this is an OpenTI Service. These services are Unisys-based OLTP transactions that are initiated from the Windows platform.

NN (char. 10-11 or 11-12): A number from 1 to 99 indicates the service number. There can be up to 99 services in a particular server.

Examples:

1. Create the first service within the second server of the AUR production environment:

   Server & Element Name (server 2, service 1): P02AURSVC01

2. Create the first service for the first server in the Unit Test environment.

   Server & Element Name (server 1, service1): U01AURSVC01

3. Create three (3) Production services. The first 2 services are for server 1 and the other for server 2.

   Server & Element Names (server 1, service 1 & 2): P01AURSVC01
   P01AURSVC02

   Server & Element Name (server 2, service1): P02AURSVC01

Notes:

1. The first twelve (12) characters of the service name and the mass storage element name are to be the same.

2. All internal/external/documentation references to a service name must be consistent with these naming standards.

3. By including the server number within the service naming convention the view numbering can begin at one (1) for each service rather the being sequentially numbered across all services and servers in order to insure uniqueness.

Naming Convention for Remote Services

Name Format: **AAANNRSVCNNOTI**

Element Format: **AAANNRSVCNNOTI**

Explanation:

AAA (char. 1-3): A three (3) character acronym for the system/subsystem. It may consist of upper case letters and numbers. It must, however, begin with a letter.

NN (char. 4-5): A number from 1 to 99. Indicates the server number. There can be up to 99 servers in a particular system/subsystem.

RSVC (char. 6-9): Indicates that this is the client of a Remote Service. These servers may be Windows-based COM+ transactions that can be initiated from OLTP transactions on the Unisys platform.

NN (char. 10-11): A number from 1 to 99 indicates the service number. There can be up to 99 services in a particular server.

OTI (char. 11-13): Indicates that this is the client of an OpenTI Server. These servers are Windows-based COM+ transactions that are initiated from OLTP transactions on the Unisys platform.

Examples:

A Unisys client to an OpenTI Server:

Server & Element Name: AUR03RSVC02OTI

Design Considerations: Servers may contain multiple services. Whether to incorporate multiple services into a single service is a design decision that requires some consideration based on the functional requirements of the application.

Example A: The Server is registered as an update transaction. This transaction has three functional requirements. Delete, Insert, and Modify. One Service 01 could be written to accommodate the transaction.

Service 1 to Server 1:

P01AURSVC01 uses P01AURSVR

Example B: The Server is registered as an update transaction. In this case, three separate Services could be written. Service 01 Delete, Service 02 Insert, and Service 03 Modify.

Services to Server 2:

P02AURSVC01, P02AURSVC02 uses P02AURSVR

Continuity of Server and Service Names:

This example flows an OLTP Service from the AUR application area.

P01AURSVR	1st Server deriving data from the AUR database. The first six characters of the server name **P01AUR** is the registered transaction within the OS2200 Validation Table or VALTAB.
P01AURSVC01	1st Service deriving data from methods using the Server component P01AURSVR.
P01AURSVC02	2nd Service deriving data from methods using the Server Component P01AURSVR.
P01AURRSVC01	1st Remote Service for AUR, which does not require 2200 VALTAB registration or Server. This remote service does require registration into the OS configuration file.

COM+ Naming Example:

P03AURSVC01OpenTI	1st COM+ Service in the third server providing data to the AUR application system. (See Service Layer Components OpenTIComponentStds.doc)

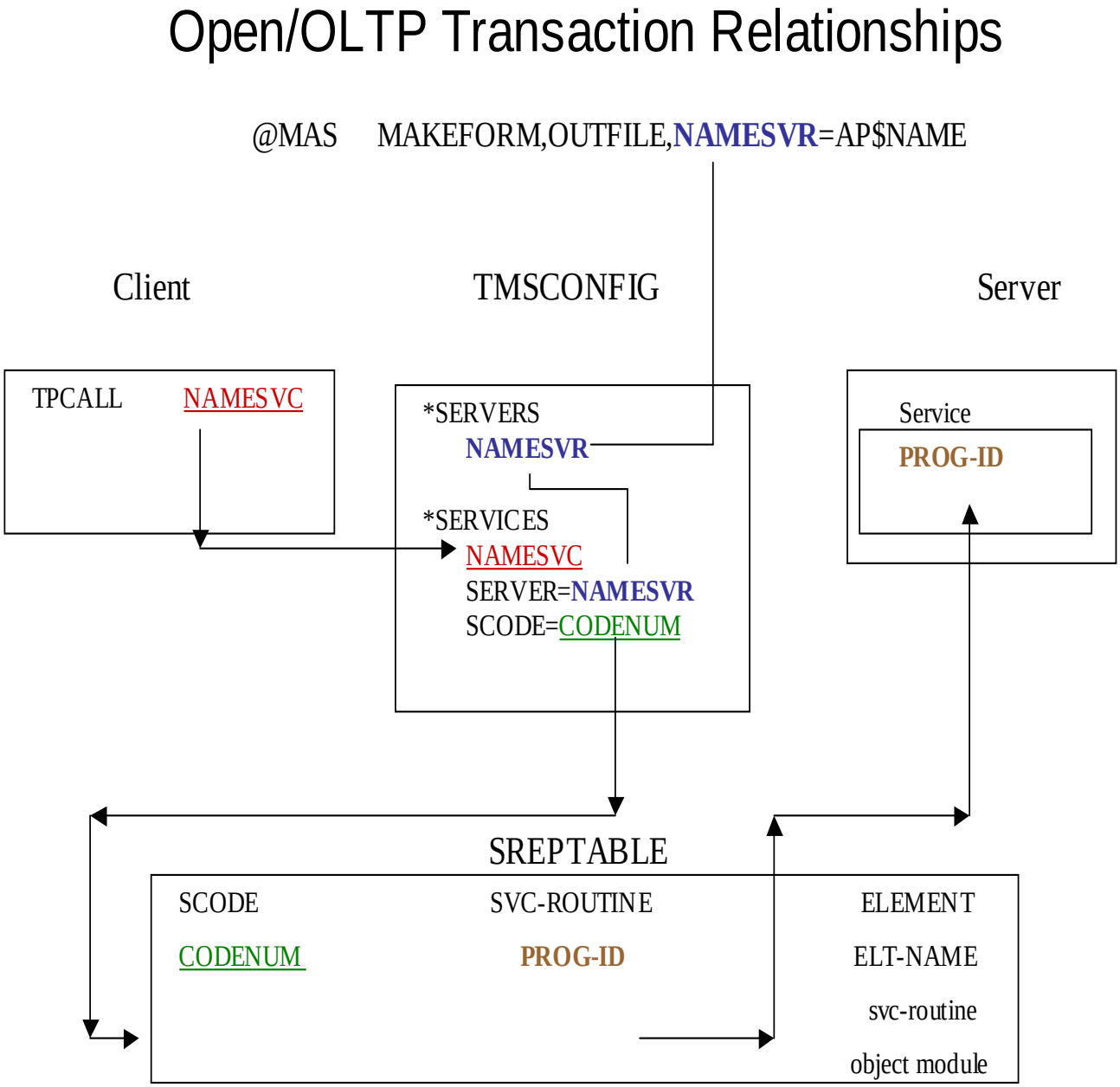
APNAME

AP_Name is used by UNISYS to describe Server and Service Names in the TMSCONFIG file.

Section	Parameter	Description
APNAME	*Ap_name*	Defines the name of the AP (client or service routine). The *apname* must be a string of 1 to 12 characters consisting of uppercase and lowercase letters, digits from 0 through 9, underscore characters (_), and dollar signs ($) in any combination. If the AP is configured in the TMSCONFIG file, the name you specify here must be identical to the configured name. Note that client programs often do not need to be configured in the TMSCONFIG file. See the *Open/OLTP Transaction Manager Administration Guide Volume 1* for more information about configuring APs. If you omit the APNAME SGS, the name defaults to AP$GENERIC.
*SERVERS	*server_name* *NAMESVR*	Specifies a unique 1- to 12-character user-defined name for the server.
*SERVERS	UDS_ACCESS UDS_RECOVERY	If the application group to which the server and service belong defines UDS as a local RM (RMID=UDS), the *SERVERS section of the TMSCONFIG file defines the following information about UDS: • Access mode (retrieval or update) • Recovery option (quicklooks or deferred).
*SERVICES	*svc_name* *NAMESVC*	Defines a unique service name up to 32 characters long (only the first 15 characters are significant). The service name is the name used by the client to request work from a specific service routine.

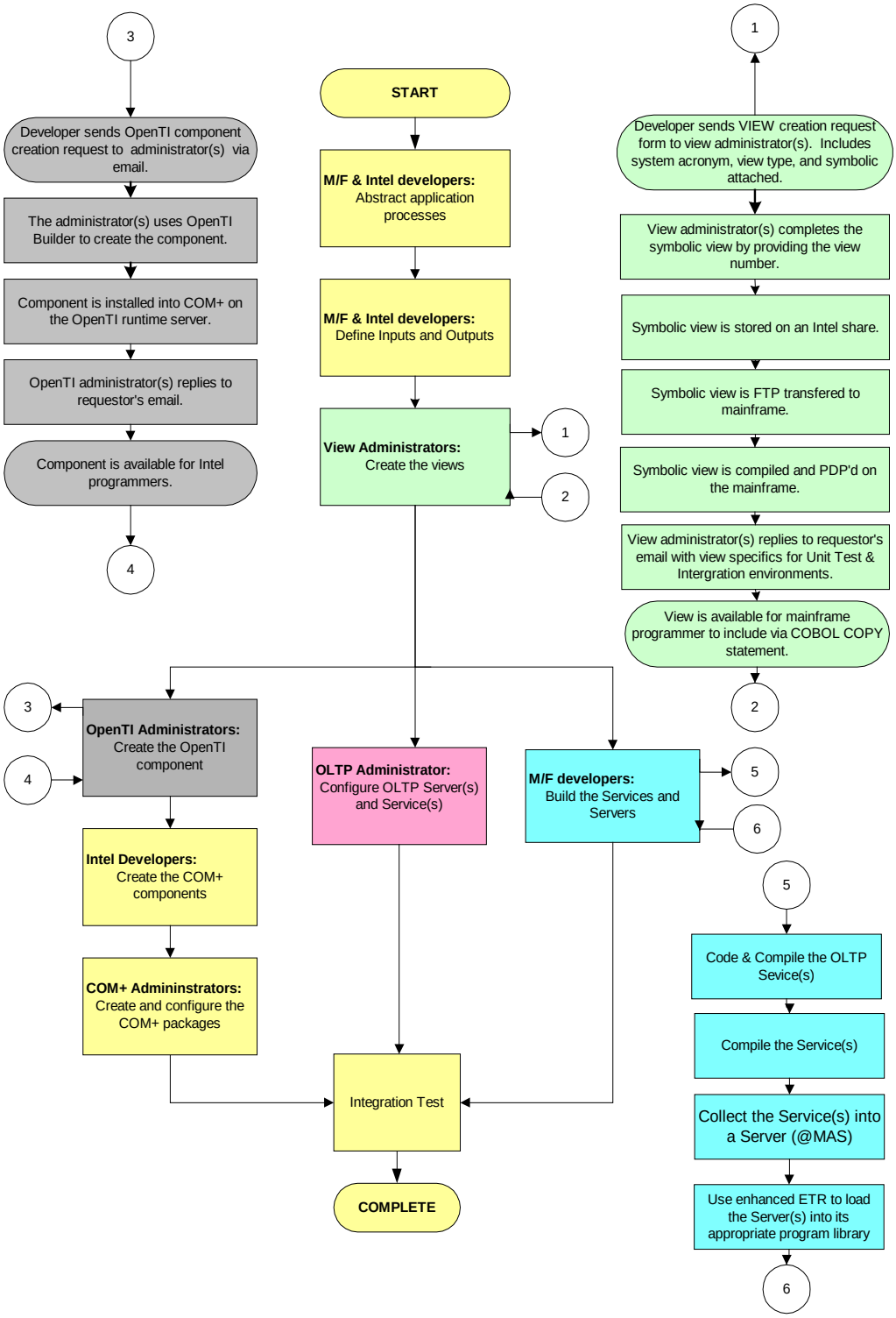
Section	Parameter	Description
*SERVICES	ACCESS	Makes a service available (ON) or unavailable (OFF) to handle requests. The tpadvertise() function sets the ACCESS parameter to ON. The tpunadvertise() function sets the ACCESS parameter to OFF.
*SERVICES	AUTOTRAN	Configures AUTOTRAN on or off. When AUTOTRAN is ON, the service is always executed in transaction mode, even if the AP that calls it is not in transaction mode or specifies TPNOTRAN on the call.
*SERVICES	BUFTYPES	Identifies the buffer types and subtypes allowed by the service.
*SERVICES	NAME	Specifies the actual service name in cases where the service name is case sensitive or contains characters not allowed for a label *(svc_nam*e). If specified, the value for NAME overrides *svc_nam*e.
*SERVICES	SCODE	Specifies the index for the service routine in the service table. This index allows the server to find the entry point for the service routine that handles the service.
*SERVICES	TRANTIME	Specifies the number of seconds allowed before an AUTOTRAN transaction becomes susceptible to transaction timeout.

Transaction Relationship Diagram



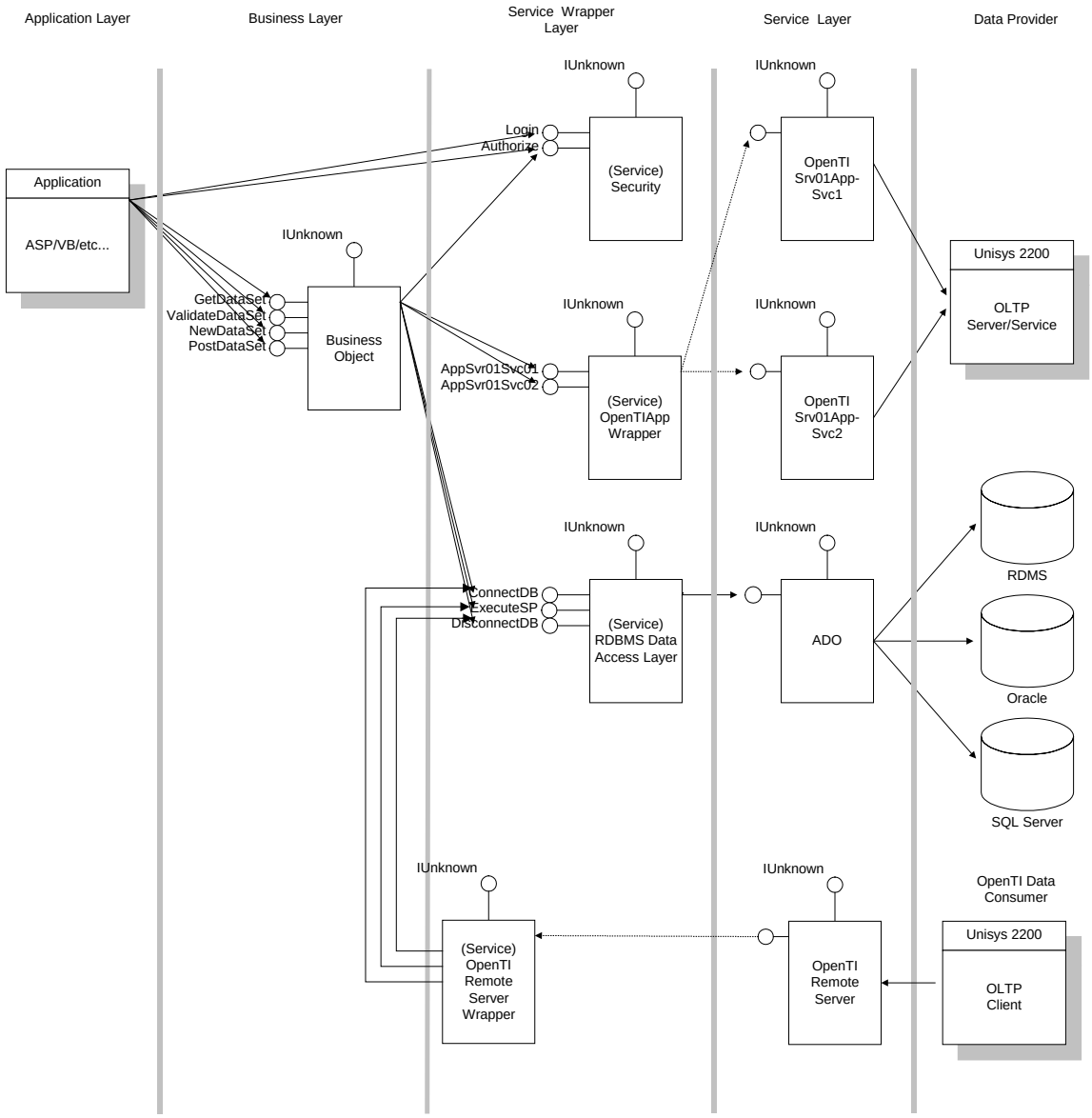
Consolidated Process Flow

Below is a diagram depicting the entire process flow for the creation of OpenTI Service Components.


3
Developer sends OpenTI component creation request to administrator(s) via email.
The administrator(s) uses OpenTI Builder to create the component.
Component is installed into COM+ on the OpenTI runtime server.
OpenTI administrator(s) replies to requestor's email.
Component is available for Intel programmers.
4
START
M/F & Intel developers: Abstract application processes
M/F & Intel developers: Define Inputs and Outputs
View Administrators: Create the views
1
2
1
Developer sends VIEW creation request form to view administrator(s). Includes system acronym, view type, and symbolic attached.
View administrator(s) completes the symbolic view by providing the view number.
Symbolic view is stored on an Intel share.
Symbolic view is FTP transfered to mainframe.
Symbolic view is compiled and PDP'd on the mainframe.
View administrator(s) replies to requestor's email with view specifics for Unit Test & Intergration environments.
View is available for mainframe programmer to include via COBOL COPY statement.
2
3
4
OpenTI Administrators: Create the OpenTI component
Intel Developers: Create the COM+ components
COM+ Admininstrators: Create and configure the COM+ packages
OLTP Administrator: Configure OLTP Server(s) and Service(s)
M/F developers: Build the Services and Servers
5
6
5
Code & Compile the OLTP Sevice(s)
Compile the Service(s)
Collect the Service(s) into a Server (@MAS)
Use enhanced ETR to load the Server(s) into its appropriate program library
6
Integration Test
COMPLETE

COM+ Component Architecture Diagram

The diagram below demonstrates, at a high level, the interaction between Applications, COM+ components and data stores. This is not intended to describe a physical topology, but rather, a logical explanation of the relationships involved. It is important to gain an understanding of this architecture in order to fully comprehend the OpenTI component standards


Application Layer
Business Layer
Service Wrapper Layer
Service Layer
Data Provider
IUnknown
IUnknown
Login
Authorize
(Service) Security
OpenTI Srv01App-Svc1
Application
ASP/VB/etc...
IUnknown
Unisys 2200
GetDataSet
ValidateDataSet
NewDataSet
PostDataSet
Business Object
OLTP Server/Service
IUnknown
IUnknown
AppSvr01Svc01
AppSvr01Svc02
(Service) OpenTIApp Wrapper
OpenTI Srv01App-Svc2
IUnknown
IUnknown
RDMS
ConnectDB
ExecuteSP
DisconnectDB
(Service) RDBMS Data Access Layer
ADO
Oracle
SQL Server
IUnknown
IUnknown
OpenTI Data Consumer
(Service) OpenTI Remote Server Wrapper
OpenTI Remote Server
Unisys 2200
OLTP Client

The preceding diagram shows the logical layering of the various constituent components. Each layer is described below:

- The Application Layer is the gateway into the flow of information. This layer interacts with the Business Objects to request information for display and to send information for persisting to a data store. The application is responsible for allowing users to login (be authenticated). Applications may provide authorization via the standard security measures.
- Employing a Business Layer provides a unified method of transferring data (regardless of provider) to the application(s). Additionally, business rules and logic are contained in a single location and are not redundantly encoded. The Business Layer interacts with the appropriate data dispensers and services. The DHS interfaces (RPCs) are protected with authorization security measures. The Business Layer is responsible for beginning, committing, and rolling back transactions that are participating in a two-phase transaction strategy (MTS/MS-DTC).
- The Service Wrapper Layer provides an indirect link to the data provider and hides much complexity from the Business Layer. It is responsible for ensuring proper participation in the two-phase transactions as well as interacting with the lower level Service Layer.
- The Service Layer provides the lowest level of access (native). This layer is either provided by the operating system or by RDBMS vendors, or in the case of OpenTI, generated using vendor specific tools. This layer interacts directly with the data stores.
- Data Providers are the native databases or foreign transfer mechanisms (to databases) that participate in the DHS enterprise operations.
- For clarity of the diagram an OLTP client (host-initiated OpenTI) is shown as a data provider, when in fact it is a data consumer. It interfaces via a Service Layer to a Service Wrapper, which includes logic and references to other Service Layer objects to satisfy the processing requirements. Note that these components will likely provide some business logic, however they are not Business Components because they are not DHS.

Service Layer Components

For the purposes of this document, only the OpenTI Service Layer Components (a.k.a. OpenTI Client Components and OpenTI Server definitions) are discussed. There are other Service Layer Components but discussion of these belongs in the Component-ware Domain.

For the Windows-initiated OpenTI scenario there are several names that must be considered. They are the OpenTI client component name, the class name, then method name and the dll name.

For the Mainframe-initiated OpenTI scenario the names that must be considered For the OpenTI server definition they are the server definition name and the OpenTI server definition dll name. For the implementation component they are the component name, the class name, the method name and the dll name.

OpenTI Client Component Naming

OpenTI Client Components are built with the OpenTI Builder provided by Unisys as part of the OpenTI Development software. The naming of these components closely follows the DHS naming standards for the mainframe objects that they interface. See "Service Names" within the OpenTI/OLTP Naming Standards portion of this document or refer to the following.

Naming Convention for OpenTI Client Components

Name Format: **ENNAAASVCNNOpenTI**

Explanation:

E (char.1): The environment within which the Server operates. There are six (6) standard environments;
U = Unit test,
I = Integration test,
T = Training,
S = Systems Acceptance Test,
F = Test For Production,
P = Production

NN (char. 2-3): A number from 1 to 99. Indicates the server number. There can be up to 99 servers in a particular system/subsystem.

AAA (char. 4-6): A three (3) character acronym for the system/subsystem. It may consist of upper case letters and numbers. It must, however, begin with a letter.

SVC (char. 7-9): Indicates that this is an OpenTI Service. These services are Unisys-based OLTP transactions that are initiated from the Windows platform.

NN (char. 10-11): A number from 1 to 99 indicates the service number. There can be up to 99 services in a particular server.

OpenTI (char. 11-12): Indicates that this is an OpenTI Service. These services are Unisys-based OLTP transactions that are initiated from the Windows platform.

OpenTI Client Class Naming

Classes are created by the OpenTI Builder application. These will be named as follows:

"cls" + Server ID + Application ID + Service ID

Where:

- "cls" is the required prefix
- Server ID is two numbers from "01" to "99" which identify the mainframe Server for an application.
- Application ID is three letters representing the Application that "owns" the services.
- Service ID is two numbers from "01" to "99" that identify the sequential service being exposed from the Application's Server by OpenTI.

An example is presented below:

"cls03AUR01"

Where "cls" is required. "AUR" is the Application ID. The "03" is the Server ID. Finally, the "01" is the Service ID.

OpenTI Client Version Number

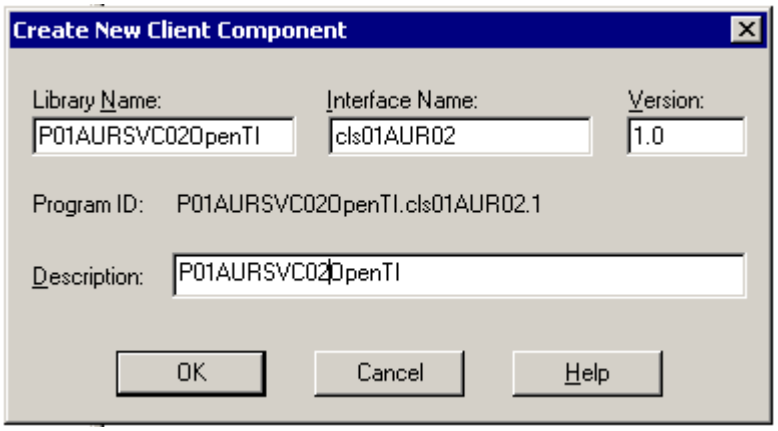
Version Number will always be 1.0. Interfaces will never be broken. If such changes must be made to an interface then a new component will be created.

OpenTI Client Description

The Description field of the OpenTI Service is used in the VB References dialog as the name of the reference. To aid in locating these references, the description field should match the Service Component Name defined earlier.

Naming OpenTI Components

The Component, Class and Version Numbers of an OpenTI Client are established in the "Create New Client Component" dialog of the OpenTI Builder as shown below.



OpenTI Client Method Naming

As there is only one method per OpenTI Component, the naming of the methods follows closely the name of the component. The naming is as follows:

Application ID + Server ID + “SVC” + Service ID

Where:

- Application ID is three letters representing the Application that “owns” the services.
- Server ID is two numbers from “01” to “99” which identify the mainframe Server for an application.
- “SVC” for Windows-initiated services.
- Service ID is two numbers from “01” to “99” that identify the sequential service being exposed from the Application’s Server by OpenTI.

An example of this is given below:

AUR03SVC01

Where the “03” indicates the Server ID. “AUR” is the application. “SVC” is always inserted to indicate a service. Finally, the “01” is the Service ID.

OpenTI Client Component File Naming

The name of the Component file must be the name of the Component with the “.dll” suffix attached.

Component Name + ".dll"

OpenTI Server Definition Component Naming

OpenTI Servers are Windows-based transactions that may be initiated from an OLTP transaction on the Unisys mainframe platform. They consist of two parts; the Server Definition and the COM+ Implementation Component. The Server Definition defines the interface to the COM+ Implementation Component, thus giving OpenTI a mechanism with which to call the COM+ Implementation Component when the request is received from the mainframe. The COM+ Implementation Component provides the server application functionality and can be called by the OpenTI run-time because it Implements the interface that is defined by the Server Definition.

This section describes the naming standards for the OLTP Server Definition .

Server Definition Naming Standard

Application Area + Server ID + “RSVC” + Service ID + “OTI”

Where:

- Application Area is three letters representing the Application that “owns” the services.
- Server ID is two numbers from “01” to “99” which identify the mainframe Server for an application.
- “RSVC” for OLTP Server (aka remote service).
- Service ID is two numbers from “01” to “99” that identify the sequential service being exposed from the Application’s Server by OpenTI.
- OTI” is used for OLTP Servers (instead of “OpenTI”) due to the limit of 15 characters for R-Name entry in the OLTP configuration.

Example for an OLTP Server in the AUR application area:

Server Definition

"AUR03RSVC01OTI"

Where "AUR" is the Application ID, "03" is the Server ID, "RSVC" indicates 'remote server' and OTI indicates OpenTI.

OpenTI Server Definition Class Naming

The class name within the OpenTI Server Definition component will be named as follows:

Application Area + Server ID + RSVC + Service ID

Where:

- Application Area is three letters representing the Application that “owns” the services.
- Server ID is two numbers from “01” to “99” which identify the mainframe Server for an application.
- “RSVC “
- Service ID is two numbers from “01” to “99” that identify the sequential service being exposed from the Application’s Server by OpenTI.

An example is presented below:

“AUR03RSVC01”

OpenTI Server Definition Version Number

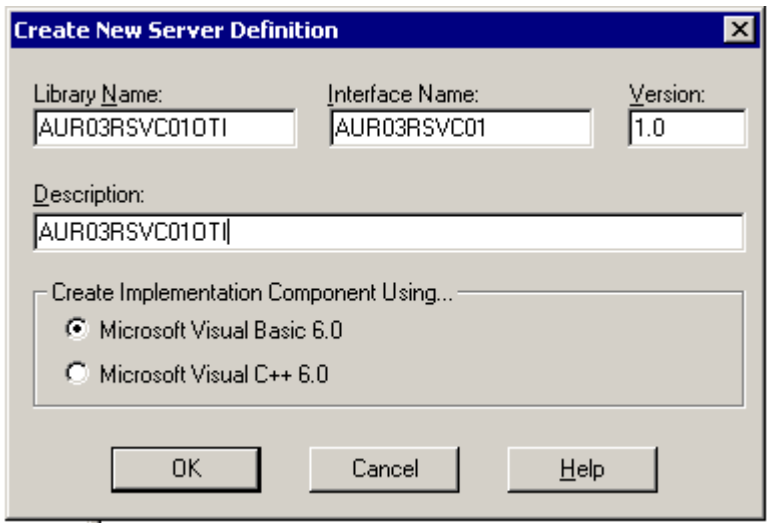
Version Number will always be 1.0. Interfaces will never be broken. If such changes must be made to an interface then a new component will be created.

OpenTI Server Definition Description

The Description field of the OpenTI Server Definition is used in the VB References dialog as the name of the reference. To aid in locating these references, the description field should match the OpenTI Server Component Name defined earlier.

Naming OpenTI Server Definitions Components

The Component, Class and Version Numbers of an OpenTI Server Definition are established in the "Create New Server Definition" dialog of the OpenTI Builder as shown below.



OpenTI Server Definition Method Naming

As there is only one method per OpenTI Component, the naming of the methods follows closely the name of the component. The naming is as follows:

Application Area + Server ID + “RSVC” + Service ID

Where:

- Application Area is three letters representing the Application that “owns” the services.
- Server ID is two numbers from “01” to “99” which identify the mainframe Server for an application.
- “SVC” for Windows-initiated services.
- Service ID is two numbers from “01” to “99” that identify the sequential service being exposed from the Application’s Server by OpenTI.

An example of this is given below:

AUR03RSVC01

OpenTI Server Definition Component File Naming

The name of the Component file must be the name of the Component with the “.dll” suffix attached.

Component Name + “.dll”

OpenTI Server COM+ Implementation Component Naming

OpenTI Servers are Windows-based transactions that are initiated from an OLTP transaction on the Unisys mainframe platform. They consist of two parts; the Server Definition and the COM+ Component. The Server Definition defines the interface to the COM+ Implementation Component, thus giving OpenTI a mechanism with which to call the COM+ Implementation Component when the request is received from the mainframe. The COM+ Implementation Component provides the server application functionality and can be called by the OpenTI run-time because it Implements the interface that is defined by the Server Definition.

This section describes the naming standards for the COM+ Implementation Component.

COM+ Implementation Component Naming Standard

"P" + RSVC" + "OpenTI" + Application Area

Where:

- "P" is used as the first character.
- "RSVC" indicates OLTP Server (formerly 'remote service')
- "OpenTI" indicates OpenTI
- Application Area is three letters representing the Application that "owns" the services.

Note that these components are not environment-specific. The leading character "P" is carried from the prior naming standard for consistency.

Also note that all the OpenTI Server COM+ methods will be accumulated into a single component for each Application Area.

Example for an OpenTI Server in the AUR application area:

Server Definition

"PRSVCOpenTIAUR"

OpenTI Server COM+ Implementation Component Class Naming

The class name within the OpenTI Server COM+ Implementation Component will be named for the corresponding server on the mainframe:

"Svr"+ Server ID

Where:

- "Svr" is the required prefix
- Server ID is two numbers from "01" to "99" which identify the mainframe Server for an application.

An example is presented below:

"Svr03"

OpenTI Server COM+ Implementation Component Method Naming

The method name for the COM+ Implementation Component is created by the OpenTI Builder. It is established on the Visual Basic code stub that is created by the builder. It is the combination of the class name and the method name of the OpenTI Server Definition. The naming is as follows:

"cls" + Application Area + Server ID + "RSVC" + Service ID

+ "_" + Application Area + Server ID + "RSVC" + Service ID

Where:

- “cls” indicates class.
- Application Area is three letters representing the Application that “owns” the services.
- Server ID is two numbers from “01” to “99” which identify the mainframe Server for an application.
- “RSVC” for OLTP Server (formerly remote service).
- Service ID is two numbers from “01” to “99” that identify the sequential service being exposed from the Application’s Server by OpenTI.

OpenTI Server COM+ Implementation Component File Naming

The name of the Component file must be the name of the Component with the “.dll” suffix attached.

Component Name + “.dll”

OpenTI Service Parameter Field Naming

OpenTI methods pass information between the Microsoft COM+ and the mainframe worlds via parameters. For the purpose of loosely coupling of components, the input and output parameters for these methods are strings. These strings are generally parsed by the Business Component that calls the service. A typical method call is described below.

```
lngReturnStatus = objOpenTI.AUR07SVC01 _
    ( _
        strInputView, _
        strOutputView, _
        lngTPurCode _
    )
```

OpenTI COM+ Application Packaging

COM+ requires that component that use it's resources be packaged into COM+ applications. While multiple components may be packed into a single application it will be our practice to package each OpenTI Service individually. The following naming conventions apply.

COM+ Application Name

The Application Name is the name assigned to the COM+ Application. It will always be the name of the Component it contains.

Component Server

The Component Server is the DNS Name of the server on which the COM+ Application will execute. OpenTI Components will run on the Intel node of the ClearPath machine. Developers

should contact the DHS Server Administrator or the OpenTI Administrator group for the DNS name of the Intel node of the ClearPath machine.

Proxy MSI Name

To access the OpenTI Service from another server, DCOM on the other server must be configured to reference it. This is accomplished by using the “COM Application Export Wizard” to export an Application Proxy (MSI file) and executing that MSI file on the other server. The MSI file must be named as follows:

COM+ Application Name + Component Server + “.MSI

Wrapper Layer Components

For the purposes of this document the only the OpenTI Service Wrapper Layer Components (a.k.a. OpenTI Wrappers) are to be discussed. There are other Service Layer Wrapper Components but discussion of these belongs in the Component-ware Domain.

OpenTI Wrapper Component Naming

OpenTI Service Wrapper Components are built in Microsoft Visual Basic. The naming of these wrapper components has been designed to facilitate relating the OpenTI components to the mainframe objects easily.

Components built to Wrap Windows-initiated OpenTI components are named as follows:

"SvcOpenTI" + Application ID

Where:

- "SvcOpenTI" is required to lead each object. In this case, Svc indicates that this is a Service Wrapper component for local (Windows-initiated) services.
- Application ID is three letters representing the Application that "owns" the services.

An example is presented below:

"SvcOpenTIAUR"

Where "SvcOpenTI" is required. "AUR" is the application.

OpenTI Wrapper Class Naming

In Visual Basic, all methods require a class module to host them. A class module can contain multiple methods. For the OpenTI Wrappers, a class is created for every Server exposed by OpenTI Service Components for an application. Thus if an application has two OpenTI servers there will be two class modules in the OpenTI Wrapper component.

OpenTI Wrapper Class Names are as follows:

"Svr" + Server ID

Where:

- "Svr" is required (to indicate Server).
- Server ID is a number from "01" to "99" matching the Server ID on the OpenTI Service Component.

An example:

Svr01

Where "Svr" is required and 01 is Server ID 01 for the hosting Wrapper Component (named after the application it is built to support).

OpenTI Wrapper Method Naming

As in Visual Basic there can be many methods in a class module, one method will be developed for each OpenTI Service Component, but all OpenTI Service Components for a given Application ID and Server ID are contained as methods in the class module.

The naming for local (Windows-initiated methods) is as follows:

"Svc" + Service ID + Logical Description

Where:

- "Svc" is a required prefix indicating a local method.
- Service ID is two numbers from "01" to "99" to match the Service ID on the OpenTI Service Method name's Service ID.
- Logical Description is a mixed case description (with no spaces) added by the Wrapper author to aid the business component developers.

An example of this is given below:

Svc01InquireRecipientInfo

Where "Svc" is required and the "01" indicates the Service ID. "InquireRecipientInfo" is a logical description of the service.

OpenTI Wrapper Method Naming Summary

To summarize the OpenTI wrapper naming conventions, an example will be used to provide a demonstration.

This example uses the P03AUR01OpenTI Service Component:

SvcOpenTIAUR->Svr03->Svc01InquireRecipientInfo

- The “Svc” indicates that this is a service or service wrapper component. The “OpenTI” indicates that the service is for OpenTI. “AUR” indicates the application name. See “OpenTI Wrapper Component Naming.”
- The “Svr” is the class name relating to a particular server. All Server ID “03” services are contained inside the class module as method names. See “OpenTI Wrapper Class Naming.”
- “Svc” indicates the particular service and initiates the method name. The “01” Service ID follows it. Finally, a logical description has been appended to complete the method name. See “OpenTI Wrapper Method Naming.”

The Visual Basic Business Object developer makes a reference to the Wrapper Component as:

```
Dim objSvcAUR03OpenTI as SvcOpenTIAUR.Svr03
```

After instantiation, access will be as:

```
objSvcAUR03OpenTI.Svc01InquireRecipientInfo _
```

```
    ( _
        strUserID, _
        strInputView, _
        strOutputView, _
        strRecipientNumber1, _
        strMedicalAssistId, _
        strRecipientName, _
        strRecipientBirthdate, _
        strRecipientSex, _
        strEligibilityFromDate, _
        strEligibilityToDate, _
        strHmoCode, _
        strHmoFromDate, _
        strHmoToDate _
    )
```

OpenTI Wrapper Parameter Field Naming

OpenTI wrapper methods pass information between itself and the Business Objects calling it via parameters. The Wrapper layer parameters mimic the names found in the Service Object, but in a slightly different format. The field Parameter names in the wrapper are preceded by the Microsoft standard data type naming convention. Additionally, the underscores are removed and Mixed case is used.

Some example parameter fields (taken from the SvcOpenTIAUR03. Svc01InquireRecipientInfo method):

strUserID

strRecipientNumber

strTextMessage

strRecipientNumber1

strMedicalAssistId

strRecipientName

strRecipientBirthdate

strRecipientSex

strEligibilityFromDate

strEligibilityToDate

strHmoCode

strHmoFromDate

strHmoToDate

An additional parameter, strUserID, is being passed into the wrapper to provide additional error trapping information.

Optionally, the input and/or output to the service wrapper may each be a string of length to match the view. In this case, the string is parsed in the Business Component.

OpenTI Wrapper Description

The Description field of the OpenTI Wrapper is used in the VB References dialog as the name of the reference. To aid in locating these references, the description field should be set as follows:

Prefix	"SvcOpenTI"
Component	ComponentName
Description	Description of Component, less than 40 characters in length.

OpenTI Wrapper Component File Naming

The name of the Component file must be the name of the Component with the ".dll" suffix attached.

Component Name + ".dll"

OpenTI Wrapper COM+ Application Packaging

COM+ requires that the component that uses its resources be packaged into COM+ applications. While multiple components may be packed into a single application it will be our practice to package each OpenTI Service Wrapper individually. The following naming conventions apply.

COM+ Application Name

The Application Name is the name assigned to the COM+ Application. It will always be the name of the Component it contains.

Component Server

The Component Server is the DNS Name of the server on which the COM+ Application will execute. OpenTI Service Wrapper Components will run on the designated Application Server machine. Developers should contact the DHS Server Administrator to identify the designated Application Server for components.

Proxy MSI Name

To access the OpenTI Service from another server, DCOM on the other server must be configured to reference it. Since Service Wrappers are intended to be invoked only by processes on the same machine, this standard does not apply.

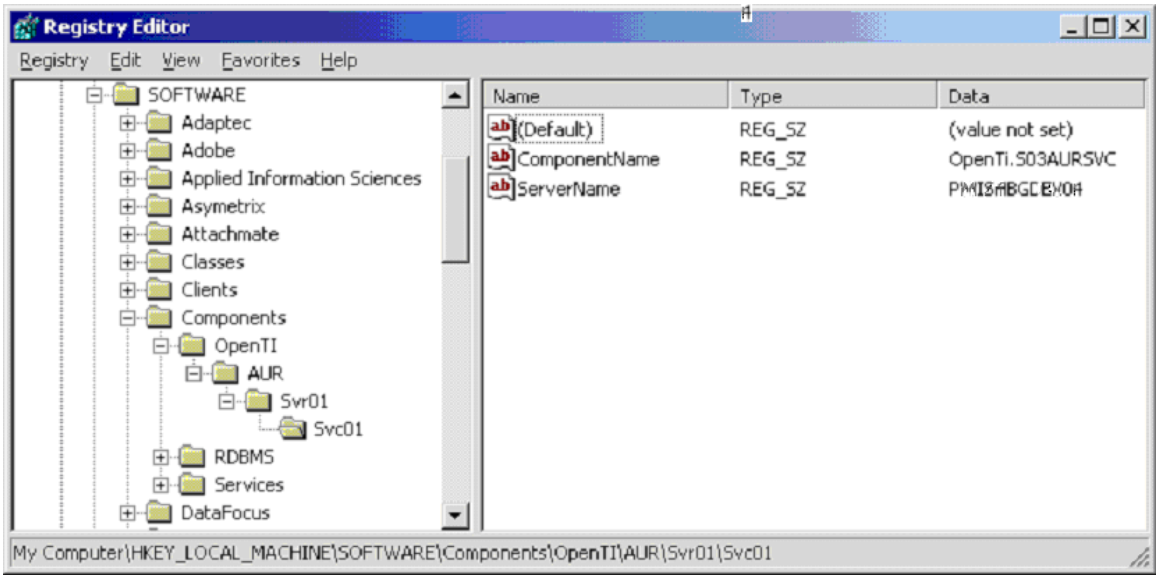
Registry Settings for OpenTI Wrapper Components

OpenTI Wrapper Components determine the OpenTI Service they are wrapping by reading the registry of the server on which the wrapper resides.

OpenTI Wrapper Component Registry Location

The name of the actual OpenTI component is stored in the system registry. The wrapper components read the registry settings for the appropriate OpenTI component prior to binding at runtime (late binding).

See the diagram below for an example of how the registry is used to store the OpenTI Service Component information:



Hive Name

The specific hive name is HKEY_LOCAL_MACHINE, this is to ensure that regardless of who logs into the server, that the proper registry location is used.

OpenTI Path

The path following the hive name is as follows:

\COMPONENTS\OPENTI

Specific OpenTI Application, Server and Service

The path following the OpenTI Path defines a specific OpenTI Service Component and is as follows:

\ApplicationID\ServerID\ServiceID

Where:

- ApplicationID is the three letter representation of the owning Application
- “Svr” is required and ServerID is a value between “01” and “99” indicating the Server ID
- “Svc” is required and ServiceID is a value between “01” and “99” indicating the Service ID.

Registry Values for OpenTI Wrapper Components

The registry location for any given OpenTI Wrapper Component contains two specific values as follows:

- ComponentName – A string value containing the name of the specific OpenTI Service Component to be accessed by the wrapper. This is required.

- ServerName – A string value containing the name of the Windows NT or Windows 2000 Server on which the OpenTI Service Component is to be executed. This is an optional setting; the default specifies the same server on which the OpenTI Service Component resides.

If the ServerName registry setting is not found, then the component will be instantiated without referencing a server name. In this case, the OpenTI component, or a proxy for it, must be installed on the machine. This is ideal for the NLB environment used in SAT and production at DHS.

OpenTI Wrapper Component Registry Coding Requirements

The wrapper must meet certain requirements to ensure uniform access of the OpenTI Services.

Constant Definition

This registry location is defined as constants in a wrapper component's .bas module as follows:

```
'
'   General OpenTI Registry Key
'
DHS Const OPENTIREG As String = _
    "\HKEY_LOCAL_MACHINE\SOFTWARE\COMPONENTS\OPENTI\"
'
'   Specific Application/Server/Service Registry Entries
'
DHS Const AUR0301SERVER As String = "AUR\SVR03\SVC01"
```

Where:

- "\HKey_local_machine\Software\Components\OpenTI\" is required
- Application is the three letter ApplicationID representation
- "Svr" is required and Server is a value between "01" and "99" indicating the Server ID
- "Svc" is required and Service is a value between "01" and "99" indicating the Service ID.

Registry Access Mechanism

Wrapper components use the SvcRegistry Component to retrieve the OpenTI Service Component information from the Registry. See the example below:

```
'
'   Get OpenTI Component Name
'
strOpenTiComponent = objRegistry.ReadRegistry _
( _
    OPENTIREG & AUR0301SERVER & "\ComponentName\", _
    "NOTFOUND" _
)
If strOpenTiComponent = "NOTFOUND" Then
    Err.Raise _
        ERRNOOPENTICOMPONENTSTRING, _
        SOURCENAME, _
        "No Component Registry Setting Found for: " & _
            AUR0301SERVER
        '
    '
End If
```

Including OpenTI into Business Components

This section details the requirements for incorporating OpenTI into Business Components.

Purposes of Business Components Relating to OpenTI

The chief purposes of Business Components are itemized below:

- Business Components provide a standard interface for applications developers. OpenTI components (via their wrappers) are accessed from the Business Components in such a way that the details of the source or target of the data are hidden from the application developer.
- OpenTI is secured by the Business Components making calls to the Security Subsystem.
- OpenTI transactions are controlled by the Business Components (MTS/MS-DTC 2 phase commit)

Business Component Naming

Business Components are built in Microsoft Visual Basic. The naming of these components is as follows:

"Bus" + Application Area

Where:

- "Bus" is required to lead each business object. Bus indicates that the object is a Business specific component.
- Application Area is the name of the application that the Business component supports.

An example is presented below:

"BusAUR"

Where "Bus" is required. "AUR" is the application.

Business Component Description

The Description field of the Business Component is used in the VB References dialog as the name of the reference. To aid in locating these references, the description field should be set as follows:

Prefix "Bus"

Component	Application Area
Description	Description of Component, less than 40 characters in length.

Business Component Class Naming

In Visual Basic, all methods require a class module to host them. A class module can contain multiple methods. For the Business Components, a single “Interface” class is created.

Business Component Class Names are as follows:

“I” + Application Area

Where:

- “I” is required (to indicate Interface).
- Application Area is the name of the application that the Business component supports. Please note that if the name is an acronym it is in all upper-case characters. If the name is spelled out it is in mixed case.

An example is presented below:

“IAUR”

Where “I” is required. “AUR” is the application.

To create this component the class must be referenced, for example to create the AUR Business Component, Dim the object as “BusAUR.IAUR”

```
Dim objIAUR as BusAUR.IUR
Set objIAUR = New BusAUR.IAUR
```

Business Component Method Types

Business Component methods (DHSly accessible) are constructed in such a way that application developers requiring the services of the business component do not need the actual implementation details. There are 4 standard method types and not all may be required in a Business Component. These are:

1. GetDataSet – This interface returns one or more record sets (formatted as XML) from the data source, in this case an OpenTI Wrapper Component. The client application converts the record set or sets from XML strings into record sets.

2. ValidateDataSet – This interface takes in a record set (formatted as XML) from the client application and checks the validity of the data. Invalid conditions are put into a validation record set, converted to XML and sent back to the client application.

3. NewDataSet – The output of this interface is an empty record set (formatted as XML) with the fields that match those produces by the GetDataSet method. This allows the client application to create an empty record set for adding records that need to be posted or for other purposes related to processing.

4. PostDataSet – This interface takes in one or more record sets (as XML formatted strings) that are then processed. The processing formats these input record sets into the format required by the OpenTI Component Wrapper.

5. IsExistingDataSet – This interface searches for the existence of a single data item of the DataSet type specified in its name. The exact item to search for is specified as an input parameter. The result of the IsExistingDataSet is a Boolean value where it is true if the requested item is found, false if not.

Alternatively, Business Component method names may be named for the specific functionality that they implement. This functionality may be a business process that is more complex than

those cited above. In this case, the method name should be as descriptive as possible of the functionality being implemented.

Business Component Method Naming

In the Business Component Method Types section above, the methods were named with "DataSet" as part of the name. This convention is common and used consistently within the Business Objects.

Business Component method names follow the following format:

(Standard Method Type Prefix)

PrefixDataset

-or-

(Non-Standard Method Type Prefix)

PrefixDescription

Standard Methods

Standard Method naming in business objects provides the base functionality of the object and usually satisfy all of the business requirements of applications.

Standard Method Prefixes

This is required for each standard method type and has one of the following four values:

Get, Validate, New, Post, IsExisting

Standard Method DataSet Name

The DataSet Name is a fictitious name given to the methods that describes the data being handled by the methods. For example, In AUR there is an OpenTI Service that returns recipient information. A DataSet name based on this information could be called RecipientInfo. Once selected, all methods that support the recipient information data will use the RecipientInfo DataSet name.

Standard Method Name Examples:

1. GetRecipientInfo
2. ValidateRecipientInfo
3. NewRecipientInfo
4. PostRecipientInfo
5. IsExisitingRecipientInfo

Non-Standard Methods

Non-Standard methods are used to expand the normal functionality of the components in which they reside to provide additional functionality of a business nature to applications.

Non-Standard Method Prefixes

Non-Standard method type prefixes are used to describe the expanded functionality of the components they reside in. A prefix is required for each non-standard method type and has one of the following values:

Is, Process

- The “Is” prefix is used to create various Boolean methods. An appropriate description of the evaluation being performed by the method will follow the “Is”

- A prefix for describing a process that does not pertain to data sets is prefixed with “Process.” The types of activities that could take this prefix include: processing of some non-formatted flat file

Non-Standard Method Descriptions

The Non-Standard methods, in addition to a prefix, require a description to fully describe their purpose. This description must be in mixed-case mode and describe briefly the target (or purpose) of the prefix to which they are attached.

Non-Standard Method Name Examples:

1. IsPersonEligible (“Is” prefix example)
2. ProcessRequestEMail (“Process” prefix example)

Business Component File Naming

The name of the Component file must be the name of the Component with the ".dll" suffix attached.

Component Name + ".dll"

Business Component COM+ Application Packaging

COM+ requires that component that use it's resources be packaged into COM+ applications. While multiple components may be packed into a single application it will be our practice to package each OpenTI Service Wrapper individually. The following naming conventions apply.

COM+ Application Name

The Application Name is the name assigned to the COM+ Application. It will always be the name of the Component it contains.

Component Server

The Component Server is the DNS Name of the server on which the COM+ Application will execute. Business Components will run on the designated Application Server machine. Developers should contact the DHS server administrator to identify the designated Application Server for components.

Proxy MSI Name

To access the OpenTI Service from another machine, DCOM on the other server must be configured to reference it. The other machine may be another application server, an IIS server, or a smart client system. This is accomplished by using the "COM Application Export Wizard" to export a Business Component Proxy (MSI file) and executing that MSI file on the other machine. The MSI file must be named as follows:

COM+ Application Name + Component Server + ".MSI

Attachment A - Forms

OLTP ADMINISTRATION REQUEST

SERVERS

REQUESTING UNIT: _____ **DATE OF REQUEST:** _____

NAME: _____ **PHONE #:** _____

E-MAIL ADDRESS: _____

HOST _____ (HSH-A, HSH-C)

SERVER NAME: _____

*** The Server name is a unique user-defined name; (Ex. U01CISSVR) . Reference the OLTP naming convention standards for SERVERS.

APPLICATION GROUP ID: _____ (UDS8, UDS1)

*** If Global transaction, The APPLICATION GROUP ID is required

UDS_ACCESS_MODE: _____ (Retrieval or Update)

*** If the Application Group Id is UDS, then UDS_ACCESS_MODE is required

UDS_RECOVERY_OPTIONS: _____ (Quicklooks or Deferred)

*** If the Application Group Id is UDS, then UDS_RECOVERY_OPTIONS is required

BATCH SERVERS

START: _____

*** If Server is a Batch program, enter Start image to start the Server (can be up to 134 characters in length)

TIP SERVERS

TRANSACTION CODE: _____

*** Transaction code used to schedule TIP Server (Ex. U03AUR). Trans. Code is restricted to 6 characters in length

*** Application group Name and Number are required for Tip Servers

APPLICATION GROUP NAME: _____ (Ex. DDMR8)

APPLICATION GROUP NUMBER: _____ (Ex. 8)

OLTP ADMINISTRATION REQUEST

SERVICES

REQUESTING UNIT: _____ **DATE OF REQUEST:** _____

NAME: _____ **PHONE #:** _____

E-MAIL ADDRESS: _____

HOST: _____ (HSH-A, HSH-C)

SERVICE NAME: _____

*** This field is required. The Service name must be unique within any Service name definition in any section of the TMSCONFIG file, and is restricted to 12 characters in length. For simplicity the Service name should match the Directory Service Name, but does not have to. The first 6 characters of the Service name must be the name of the Server

that owns it. Example (U01CISSVC01) Reference the OLTP naming standards for Services.

DIRECTORY SERVICE NAME: _____

*** The name must be unique and is restricted to 12 characters in length. The name specifies the actual service name to be stored in the OLTP_TM2200 directory services database. This is the name the application program will specify on the TPCALL( ), TPACALL( ), or TPCONNECT( ). If no name is specified, the Service Name value is converted to uppercase and stored as the directory service name.

SERVER NAME: _____

*** The name of the server that owns the service. The value must be a server name defined in the Servers section, and must be 6 characters in length.

BUFFER TYPE: _____ (X_COMMON, X_OCTET, X_C_TYPE)

*** Up to 3 Buffer types and 10 Subtypes (views) can be specified for each service. X_COMMON and X_C_TYPE require user-defined Subtypes (views) to be created.

SCODE: _____ (Default = 0)

*** This field is required. The SCODE is an index for the service routine in the service table. The value must be an integer greater than or equal to zero. It corresponds to the service routine table of the MAS (Make a Server) make-form input.

OLTP ADMINISTRATION REQUEST

REMOTE SERVICES

REQUESTING UNIT: _____ **DATE OF REQUEST:** _____

NAME: _____ **PHONE #:** _____

E-MAIL ADDRESS: _____

HOST: _____ (HSH-A, HSH-C)

SERVICE NAME: _____

*** This field is required. The Service name must be unique within any Service name definition in any section of the TMSCONFIG file, and is restricted to 12 characters in length. For simplicity the Service name should match the

Directory Service Name, but does not have to. The first 6 characters of the Service name must be the name of the Server which owns it. Example U01CISRSVC01 Reference the OLTP naming standards for Remote Services.

DIRECTORY SERVICE NAME: _____

*** The name must be unique and is restricted to 12 characters in length. The name specifies the actual service name to be stored in the OLTP_TM2200 directory services database. This is the name the application program will specify on the TPCALL( ), TPACALL( ), or TPCONNECT( ). If no name is specified, the Service Name value is converted to uppercase and stored as the directory service name.

REMOTE SERVICE NAME: _____

*** The name of the remote service, 1 to 15 characters in length. The name must be enclosed in double quotes. If the Remote Service Name is omitted the default is the Directory Services Name.

AE_ALIAS: _____

*** Name 1 to 16 characters in length, which identifies a remote machine. The name must match a valid remote site name in the AE_ALIAS field of the CONFIG/IN element in the OSI-TP configuration file.

OLTP ADMINISTRATION REQUEST

VIEWS

REQUESTING UNIT: _____ **DATE OF REQUEST:** _____

NAME: _____ **PHONE #:** _____

E-MAIL ADDRESS: _____

HOST _____ **(HSH-C, HSH-A)**

SERVER ASSOCIATED WITH VIEW: _____

Name of Server (from Server request form) associated with the view

SERVICE ASSOCIATED WITH VIEW: _____

Name of Service (from Service request form) associated with the view

VIEW NAME: _____

*** The View name is a unique user-defined name; (Ex. VAUR0000020I). Reference the OLTP naming convention standards for Views. All view names will begin with a V for view, a 3 digit application name, the length of the view, and **I**nput/**O**utput. Reference the View naming convention in the OpenTI standards.

FILE/ELEMENT containing view on OS2200: _____

(Location of the view source. Ex. OLTP*MYFILE.VAUR0000020I/VIEW)

COBOL COMPILE (OLTP/HSH-C): _____

**** For Mainframe developers only. Is your program compiled using ACOB or UCOB?

VIEW REGISTRATION:

Place check mark beside area where you want view registered. ***Element Transfer (ETR) to be used for Production**.**

OLTP (HSH-C): ____

OPENTI: (Server) ____

VIEW STATUS: _____ (NEW or UPDATE)

Is this a NEW view, or an UPDATE to an existing view

Completed form should be returned or e-mailed to your View Administrator

OLTP ADMINISTRATION REQUEST

Event/User Logs

REQUESTING UNIT: _____ **DATE OF REQUEST:** _____

NAME: _____ **PHONE #:** _____

E-MAIL ADDRESS: _____

HOST: _____ (HSH-A, HSH-C)

There is a maximum of 10 Event Log configuration entries per system.

EVENT_TYPE: _____

*** This is a required field, The Event_Type specifies the types of events which are written to the event Log. Multiple event types may be specified

EVENT_TYPES	Description

OLTP_REMARK	Specifies that all low-severity events detected by OLTP-TM2200 are written to the user-defined log, in addition to the OLTP-TM2200 system event log.
OLTP_WARNING	Specifies that all medium-severity events detected by OLTP-TM2200 are written to the user-defined log, in addition to the OLTP-TM2200 system event log.
OLTP_ERROR	Specifies that all high-severity events detected by OLTP-TM2200 are written to the user-defined log, in addition to the OLTP-TM2200 system event log.
OLTP_ALARM	Specifies that all alarm events (requiring operator or administrative intervention) detected by OLTP-TM2200 are written to the user-defined log, in addition to the OLTP-TM2200 system event log.
OLTP_EXTERN	Specifies that all events logged by external products using the HAA access interface (for heritage application programs) are written to the user-defined log, in addition to the OLTP-TM2200 system event log.
OLTP_USER	Specifies that all events logged using the userlog() function are written to the user-defined log.
*"type-nam*e"	Specifies the name of a user-defined event type. The name must be unique and can be up to 12 alphanumeric characters in length, enclosed in double quotation marks. There can be a maximum of 10 different user-defined event types for all event logs.

*** If an event log is not defined for the OLTP_USER event type, all events are written to the standard output stream (PRINT$).

OUTPUT_DEVICE: _____

Keyword	Description
CONSOLE	Specifies that the events are written to the OS 2200 system operator console.
STDOUT	Specifies that events are written to the standard output for the activity logging the event.
ALARM_MONITOR	Specifies that events are written to standard Simple Network Management Protocol (SNMP) products as alarm messages.
ERROR_MONITOR	Specifies that events are written to standard Simple Network Management Protocol (SNMP) products as error messages.
INFO_MONITOR	Specifies that events are written to standard Simple Network Management Protocol (SNMP) products as informational messages.
FILE	Specifies that events are written to a mass storage file.

*** The following parameters are ignored of the OUTPUT DEVICE TYPE is not FILE

File_Control: _____

The full name of the file to be catalogued

File_Descriptor: _____

A unique identifier for referring to the file from the operator console interface.

The descriptor must be 1 to 6 characters long. If name omitted file cannot be cycled from console.

Event_Queue_size: _____

The maximum number of messages which can be buffered for the file, the default is 100

Department of DHS Welfare, Office of Information Systems

TRANSACTION REGISTRATION COVER

The information entered on this cover form must apply to all individual registration forms attached.

Item Name	Description/Instruction
Number of individual forms submitted with this cover	Enter the number of individual transactions attached to this cover page.
Application Unit:	
Unit Name:	Enter the name of the unit submitting the registration form.
Project Name:	Enter the name of the project associated with this (these) transaction(s).
Analyst Name:	Enter the name of the analyst assigned to this (these) transaction(s).
Action:	Check the appropriate type of action to be taken for this (these) transaction(s).

Security:	Transaction Security numbers will be assigned by BTE staff. OLTP and OpenTI do not require Security registration numbers at this time.
VALTAB:	Register Screen names, If transaction is inquiry or update mark the appropriate nodes.
WebTS:	Indicate if transaction is web enabled.
Capacity:	Enter Bank size for I-Bank, D-Bank and Common-Bank.
OLTP / OpenTI:	Indicate View names, if Server or Service is an Update or Inquiry. Mark appropriate Node. Register Server for each environment separately.
Program Office Security Authorization:	Indicate the person authorizing the transaction, the method in which the authorization was received (phone call, meeting, etc.), and the date on which the authorization was given **OR** if the authorization has been secured through a separate document, enter a checkmark on the form and attach the document approving the transaction(s).
BOB Registration:	An original signature must be secured and dated for the Bureau of Budget staff person registering the transaction(s) and verifying the correctness of the accounting information.
BIS Authorization:	An original signature must be secured and dated for the DSE staff person approving the transaction(s).

Department of DHS Welfare, Office of Information Systems

TRANSACTION REGISTRATION COVER

The information entered on this cover form must apply to all individual registration forms attached.

(For Division of Systems Engineering and Enterprise Operating Systems Section Use Only)

Security:	To be signed and dated by the DSE staff person completing the activity related to registering the transaction(s).
VALTAB:	To be signed and dated by the DSE staff person completing the activity related to assigning a VALTAB(s) to the transaction(s).
Web TS Registration:	To be signed and dated by the DSE staff person completing the activity related to enabling the Web registration.
Capacity:	To be signed and dated by the DSE staff person completing the activity related to capturing capacity information.

OLTP/ OpenTI: To be signed and dated by the DSE staff person completing the activity related to enabling the OLTP/OpenTI Server registration. The Server is registered as a VALTAB. Inquiry and update servers will be registered in separate Nodes.

Department of DHS Welfare, Office of Information Systems

ONLINE TRANSACTION REGISTRATION FORM

An individual registration form must be completed for each transaction and the information must apply to the corresponding cover form.

ITEM NAME	DESCRIPTION/INSTRUCTIONS
Transaction Name:	Enter the six-character transaction ID to be used in production mode.
Program / Absolute Name:	Enter the six-character name used for both the program name and the absolute name when the program is copied to the online file. This program name is not unique and may be used with one or more transaction codes to obtain different program options.
Account Code:	Enter the appropriate account code to be used for the processing of this transaction.
Name of Programmer: (Optional)	Enter the name of the programmer assigned to this (these) transaction(s). This item is optional.
Description of the Program:	Provide a complete description of the transaction being registered including the manual in which this transaction is documented for the user. Attach a copy of this material.

Host Computer(s): Check the appropriate host computer to be used for the transaction(s).

DPS: Check yes if the transaction is to have DPS. If no is

(Display Processing Screen) checked, special approval must be secured from DSE.

SUPUR File Number: Specify the transaction category to which this transaction belongs. Contact the Division of Systems Engineering if you require a new transaction category code.

Transaction Function: Check applicable box(es).

Application Group: Check the appropriate Application Group to be used for the transaction(s).

DMR1 = Production

TDMR7 = System Test

DDMR8 = Unit Test

WEB TS: Check Yes, if the transaction is to be web enabled.

Department of DHS Welfare, Bureua of Information Systems

ONLINE TRANSACTION REGISTRATION FORM

An individual registration form must be completed for each transaction and the information must apply to the corresponding cover form.

ITEM NAME	DESCRIPTION/INSTRUCTION
	Non-Security Option #1 Security Model.
Node:	***For Hosts A and D the only allowable node is 1. For Hosts A and D, the following nodes are allowable:***
	(Note: no new transactions should be placed in 1 PRDTIP).

#	NAME	#	NAME	#	NAME	#	NAME
1	PRDTIP	10	RJERUN	19	PVRUPD	28	LIHTRN
2	SPLMAP	11	SYMRUN	20	PVRINQ	29	IEVTRN
3	TDBTRN	12	RSITST	21	NURTRN	30	APPNET
4	NOTIN4	13	RJETST	22	PAHPAR	31	SIGNON

5	ICTRNS	14	SYMTST	23	REFTRN	32	AECRV1
6	BCSTIP	15	ADIUPD	24	CASTRN	33	AECRV2
7	MAPPER	16	ADIINQ	25	CISUPD	34	AECINQ
8	TIPTST	17	ADIRUP	26	CISINQ	35	AECUPD
9	RSIRUN	18	ADIRIQ	27	ADITRG	39	OTIINQ
						40	OTIUPD

Security Requirements: Indicate the appropriate type of security required for this transaction. If "other," specify the special requirements.

Note: The security requirement requested must be as specified in the Program Office Security Authorization.

SUPUR File Number: Specify the transaction category to which this transaction belongs. Contact the Division of Systems Engineering if you require a new transaction category code.

Transaction Function: Check applicable box(es).

Inquiry or Update

Transaction Implementation: VALTAB Options:

Department of DHS Welfare, Bureau of Information Systems

ONLINE TRANSACTION REGISTRATION FORM

ITEM NAME **DESCRIPTION/INSTRUCTION**

If standards are not applicable, special approval must be requested. Written requests are to be

sent to Section Chief, Enterprise Operating Systems Section, Division of Systems Engineering (DSE), Room 20,

WOB.

Program Type: The standard is Reentrant, if this is not appropriate for this transaction; the type of program must be contained in the written request to DSE and must be approved by DSE.

The program types that need DSE approval are:

Self-initializing

Self-destructive

Online Batch

High Volume program environment

Other, specify.

Program Options: The standard options set for all TIP transactions are L, N, R, and Z.

L - Permit TIP logging for this program.

N - Program must schedule another user or send an output response before terminating.

R - Program is allowed to release COMPOOL blocks.

Z - Program will produce dump upon abnormal termination.

All other options listed below require DSE approval.

Program Options:

I - FCCSS change (CG) allowed.

J - Not presently in use.

K - Not presently in use.

M - Allow maximum pages to be exceeded.

O - Test Mode.

P - Program is allowed to manipulate its SLOP table entry.

Department of DHS Welfare, Bureau of Information Systems

ONLINE TRANSACTION REGISTRATION FORM

ITEM NAME	DESCRIPTION/INSTRUCTION
Program Options:	Q - Allow manipulation of KNOS security directory.
(cont'd)	S - Program is allowed to write into protected area.
	T - Program is allows to schedule other users.
	U - FCSS release (RE) allowed.
	V - Program must submit physical COMPOOL requests.
	W - Not presently in use.
	X - Program is allowed to access protected system files.
	Y - Training mode.
Print Queue Device:	TIPPERS (TIP Production Error Queue) is the standard print queue device. If a print queue device is needed for production programs, special permission must be granted from the Division of Systems Engineering.
I/O Profile:	The standard I/O profile is database. Indicate if the transaction will be Read, Write or both. If I/O profile is non-data base, written approval is needed from DSE.

NOTE: Non-data base transactions submitted for DSE approval must contain the following information.

Metron:

Indicate whether or not the program used the Metron feature to avoid internal assignments for executing files.

Metron is for inquiry mode only. Updates must be done in Batch mode.

Non-Data Base Production

File Names:

If this TIP Transaction is a non-data base program, specify all the production file names that this specific TIP transaction will be using when put into production status.

Capacity Planning:

Complete the appropriate information for the transaction being registered.

For the size of C$SMC enter the appropriate number or name. If the number is 23 or above, special permission must be secured from DSE.

Department of DHS Welfare, Bureau of Information Systems

TRANSACTION REGISTRATION COVER

The information entered on this cover form must apply to all

individual registration forms attached.

The number of individual forms submitted with this cover:
Application Unit: Unit Name: Project Name: Analysts Name:
Action: **Security:** Add: Delete: Change: VALTAB: WebTS: Capacity: OLTP / OpenTI:
Program Office Security Authorization:

Transaction security has been approved by:

Name

Via on OR ☐ Authorization is attached.

(phone, meeting, E-Mail, etc.) Date:

Bureau of Budget (BOB) Registration:

Signature Date:

Office of Information Systems (BIS) Authorization:

Signature Date:

(For the Bureau of Technology Engineering)

VALTAB:

Signature Date:

WebTS Registration:

Signature Date:

Capacity:

Signature Date:

OLTP / OpenTI:

Signature Date:

ONLINE TRANSACTION REGISTRATION FORM

An individual registration form must be completed for each transaction and the information must apply to the corresponding cover form.

Transaction Name: _____ Program/Absolute Name:

Account Code: _____ Name of Programmer:

Description of the Program:

Host Computer(s):	**Display Processing Screen**	**Application Group:**	**WebTS**	**Open/TI**	**NODE:**
HSH-A: _____	Yes: _____	1 DMR1	_____	Yes: ☐	Yes: ☐
	_____	7 TDMR7			
HSH C: _____	No: _____	8DDMR8	_____	No: ☐	No: ☐

If No, special approval is needed

OLTP/OpenTI View Name(s):

Security Requirements: **Transaction Function:**

This application is to be added (select one) Inquiry: ☐

to the SUPUR File Code: _____ Update: ☐

Transaction Implementation VALTAB Options:

If standards are not applicable, special approval must be requested. Written requests are to be sent to Bureau of Technical Engineering Services (BTE). See instructions for information to be submitted with request.

A. Program Type: Program standard is Reentrant, if other, written approval needed.

B. Program Option: Standards are LNRZ, if other, written approval needed.

C. Print Queue Device: Standard is TIPERS, if other, written approval needed.

D. I/O Profile: Data Base ------ Read o Write π Both θ

If non-database, written approval needed.

Capacity Planning:

_____ Number of executions per day (in production mode.)

_____ Average size of transaction input in ASCII characters.

_____ Average size of transaction output in ASCII characters.

_____ Turnaround time in seconds (ever transmit to receive).

_____ Program D Bank _____Program I Bank.

_____ Size of C4SMC _____ element.

(number or name)

Attachment B – OpenTI Checklist Example

OpenTI Remote Server Checklist

Unisys 2200

OLTP Service

COBOL Program	☐	P01AURRSVC01
Input View	☐	VAUR0001000I
Output View	☐	VAUR0001000O
Service Name	☐	P01AURRSVC01
R-Name	☐	AUR01RSVC01OTI
AE_Alias	☐	OpenTIServer

Microsoft Windows

OpenTI Server Definition

Component (Library) Name	☐	AUR01RSVC01OTI
Class (Interface) Name	☐	clsAUR01RSVC01
Version Number	☐	1.0
Component Description	☐	AUR01RSVC01OTI
Input View File Name	☐	VAUR0001000I.v
Output View File Name	☐	VAUR0001000O.v
Method Name	☐	AUR01RSVC01
Server Definition File Name	☐	AUR01RSVC01OTI.dll
VB Code Stub File Name	☐	clsAUR01RSVC01_vbcode.txt

Service Wrapper Component

Component Name	☐	PrsvcOpenTIAUR
Class Name	☐	Svr01
Method Name	☐	clsAUR01RSVC01_AUR01RSVC01
Component File Name	☐	PrsvcOpenTIAUR.dll
Server Definition File Name Reference	☐	AUR01RSVC01OTI.dll

Service Wrapper COM+ Application

Application Name	☐	PrsvcOpenTIAUR
Component Server (Application Server)	☐	ComAppServer
Proxy MSI Name	☐	PrsvcOpenTIAUR.ComAppServer.msi
Proxy Server (OpenTI Server)	☐	OpenTIServer

OpenTI Local Server Registration

Server Definition File Name	☐	AUR01RSVC01OTI.dll
Server Name	☐	AUR01RSVC01OTI
Description	☐	AUR01RSVC01OTI
COM+ Application Name	☐	PrsvcOpenTIAUR
Component Name	☐	PrsvcOpenTIAUR.Svr01
Method Name	☐	AUR01RSVC01
Service Name	☐	AUR01RSVC01OTI
Description	☐	AUR01RSVC01OTI

Business Component Test Program

Project Name .....☐ TestPrsvcOpenTIAUR

Document Change Log

Change Date	Version	CR #	Change Description	Author and Organization
05/09/01	1.0		Initial creation.	DPW Advanced Technology Competency Section
02/07/03	1.1		Style edited.	Pamela Skelton
06/04/04	2.0	0497	Content Changes	Carla Solomon BADD
12/27/04	2.1		Content change to view names in View Registration Form & Attachment B	Carla Solomon Advanced Technology Competency Unit
2/25/2016	2.1		Changed DPW to DHS.	Robert Ziegler MSU