INGLENET
business solutions

TIP Internet Connector

(TIP/ic)

IP-643

This edition applies to TIP Studio 2.5 and revision levels of TIP Studio 2.5 until otherwise indicated in a new edition. Publications can be requested from the address given below.

Inglenet Business Solutions Inc reserves the right to modify or revise this document without notice. Except where a Software Usage Agreement has been executed, no contractual obligation between Inglenet Business Solutions Inc and the recipient is either expressed or implied.

It is agreed and understood that the information contained herein is **Proprietary** and **Confidential** and that the recipient shall take all necessary precautions to ensure the confidentiality thereof.

If you have a license agreement for TIP Studio or TIP/ix with Inglenet Business Solutions Inc, you may make copies of this documentation for internal use. Otherwise, you may not copy or transmit this document, in whole or in part, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of Inglenet Business Solutions Inc.

Toll Free:	1-800-387-9391
Website:	http://www.Inglenet.com
Help Desk:	HelpDesk@Inglenet.com

TIP Studio, TIP/ix, and TIP/30, and are registered trade marks of Inglenet Business Solutions Inc:

This documentation occasionally makes reference to the products of other corporations. These product names may be trade marks, registered or otherwise, or service marks of these corporations. Where this is the case, they are hereby acknowledged as such by Inglenet Business Solutions Inc.

Contents

Introduction

Inglenet Business Solutions is a software developer of network-based solutions for corporations. Web applications (e-commerce, e-business) form an important part of the service offering by Inglenet to its clients.

In addition to Web applications, Inglenet Business Solutions has a long track record of providing large-scale mission-critical online transaction processing (OLTP) applications to corporations worldwide. This large customer base relies on traditional applications running on Inglenet Business Solutions' TIP systems for various platforms.

TIP/ix is a transaction processing management system specifically designed to support the use of Unisys System 80 and 2200 Series applications in a UNIX client/server environment. Most existing TIP OLTP applications rely on the business logic written in COBOL and using TIP API. Majority of these Inglenet customers made large investments in existing solutions that have been in production for years and have long past their testing phase. Some systems have been migrated from the legacy TIP/30 environment to newer UNIX/LINUX platforms. All these customers, experiencing a stable production environment, are reluctant to discard existing systems for a more modern solution, one that would provide connectivity to the Web. However, many customers are beginning to recognize the need for integration of new Web applications with existing TIP back-end systems.

TIP Internet Connector is an optional TIP component written for Apache World Wide Web server, implemented as a FastCGI application exposing existing TIP OLTP applications to the WWW. It is running on Red Hat LINUX and AIX platforms. It requires TIP/ix running on a UNIX platform as a TIP back-end system. Communication between TIP Internet Connector and TIP/ix is provided via the TCP/IP network.

One important aspect of the TIP Internet Connector is that it generally does not require changes to the back-end TIP applications. The conversion from the screen format based MCS protocol to the HTTP, and HTML data sent to the browser, is based on a pair of files: HTML template (.htx) and TIP Internet Connector file (.t3i), containing COBOL oriented data description. Conversion of MCS screen formats into .t3i and .htx files can be done at run-time, or through the use of a tool provided by Inglenet.

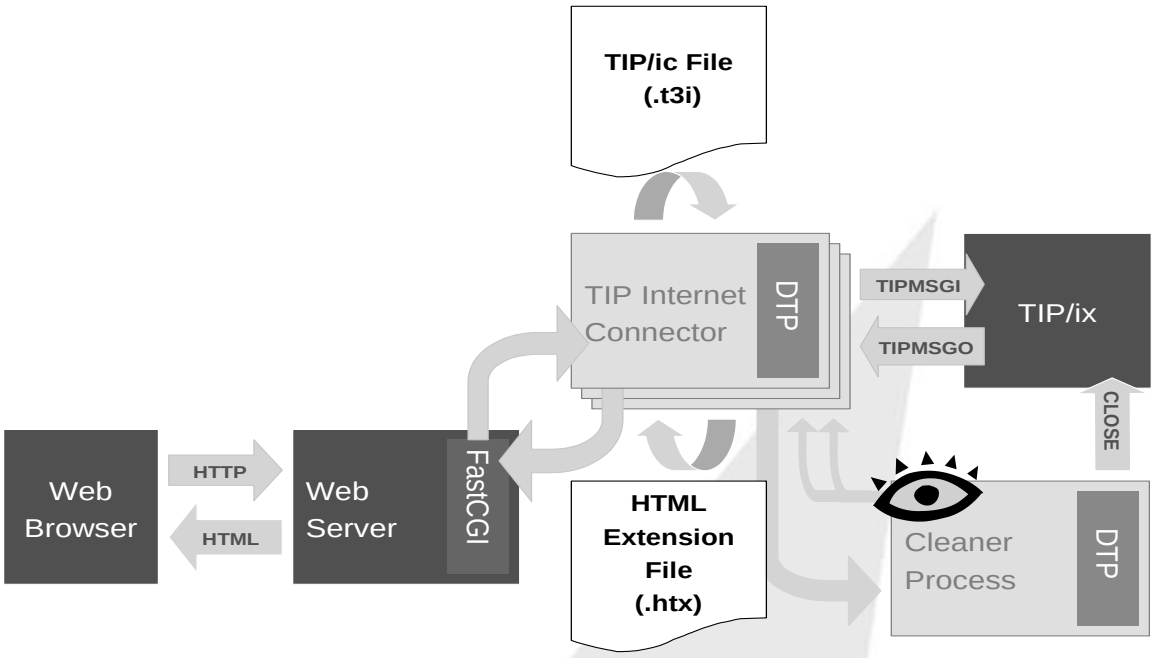
1.1 TIP Internet Connector Design Overview

TIP Internet Connector integration with the Apache Web server is done through the Fast CGI. Fast CGI was chosen as a balanced solution, with better portability than Server APIs (Apache modules), while still offering acceptable performance.

TIP Internet Connector runs single threaded, and the Web server should be configured to maintain a pool of TIP Internet Connector FastCGI processes. If an instance of TIP Internet Connector is busy processing a request, new instance will be created. Idle instances of TIP Internet Connector expire after a configurable amount of time. The number of instances running and their expiry time are all part of the Apache configuration. For a Web server to maintain a pool of the TIP Internet Connector process they have to run locally: so TIP Internet Connector version 1 is designed to run on the same machine as the Web server.

Idle instances of TIP Internet Connector can also expire after a configurable timeout in the TIP Internet Connector cleaner process. Every new instance of TIP Internet Connector checks for the existence of the cleaner process: if one is not running a new process gets started. Cleaner process periodically checks timestamps on existing sessions. As one times out it kills that instance, freeing all resources allocated by the process. Eventually, cleaner process can itself expire, freeing all resources used by TIP Internet Connector system wide.

Each instance of TIP Internet Connector maintains its own DTP connection to the TIP/ix server. DTP sessions are multiplexed across these pools of connections. No multiple responses are allowed to a single request. This is true only for MCSO input/output messages and not for DTP session and transaction messages such as OPEN, PRST, PREN).



Some changes were made on the TIP/ix side to support this simple TIP Internet Connector design.

Fig. 1. Role of the TIP Internet Connector in the system

2. Requirements for TIP Internet Connector

2.1. Operating System

TIP Internet Connector is running on Linux and AIX systems. As of February 2010, it has been tested on RedHat ES 3 & 5 as well as AIX 5.2.

2.2. World Wide Web Server

TIP Internet Connector is a FastCGI application. Although it should be able to run under other WWW servers that supports FastCGI, it was only tested with the Apache WWW server. To provide FastCGI support necessary for correct operation of the TIP Internet Connector, Apache server should be version 2.2 or newer.

Apache can be downloaded from http://httpd.apache.org/

The FastCGI API Library can be downloaded from www.fastcgi.com

The FastCGI Apache Module can be downloaded from http://httpd.apache.org/mod_fcgid/

Apache 2.2.x for AIX can also be downloaded as RPMs at http://www.perzl.org/aix/. Due to module dependencies you may have to download and install (via rpm -i) several RPM packages.

2.3. TIP Back-End System

TIP Internet Connector requires TIP/ix as a TIP back-end system. TIP Internet Connector relies on TIP/ix support for DTP connections multiplexed over several sockets. The minimal version of TIP/ix that TIP Internet Connector can connect to as a TIP LOCAP is 2.5 R0 - 0169.

It is not required that the TIP/ix system is running on the same machine as TIP/ic. TIP/ic provides interface to the TIP/ix system running on any supported UNIX platform, via TCP/IP and the proprietary DTP protocol.

There are some further limitations on the type of TIP applications that can be exposed to the WWW via TIP Internet Connector.

TIP Internet Connector 1.0 supports MCS screen format based TIP API calls TIPMSGO, TIPMSGE and TIPMSGI only. It converts MCS screen format messages to the HTML data sent to the browser.

No multiple responses are allowed on a single request (i.e. a single TIPMSGO call has to be followed by a single TIPMSGI call: no subsequent TIPMSGO or TIPMSGE calls are allowed).

Plain UTS output messages are allowed, but are ignored and never sent to the browser. TIPTERM PUT will not affect the screen flow, but the user will never see it.

3. TIP Internet Connector Configuration

3.1. Installing TIP Internet Connector

TIP Internet Connector consists of a single binary, tipic.

tipic binary should be installed and registered as a FastCGI application to the Apache Web server, as described in section 3.2.

The default configuration file should be updated with a system specific information and preferably copied to the same directory where Apache, and possibly other, configuration files are kept.

MCS screen formats can be converted to the .htx/.t3i files dynamically at run-time, or using a simple tool that shares the same MCS Screen Format Conversion library with TIP Internet Connector.

TIP Internet Connector requires that the following files be installed on the system. Locations depend on the system configuration.

file name:	description:	location:
tipic	TIP Internet Connector application	/usr/local/tipic/bin
tipic.conf	Default configuration file	/usr/local/tipic/conf
convt3z	MCS Screen Format conversion tool	/usr/local/tipic/bin
tipicconfig	tipic configuration application	/usr/local/tipic/bin

TIP Internet Connector also relies on the existence of some directories. An example is given below. The exact locations depend on the TIP Internet Connector and system configuration.

directory (example):	description:
/usr/local/tipic/	TIP Internet Connector root
/usr/local/tipic/cache	Location for dynamically created .htx and .t3i files (HTML templates and data descriptions, respectively)

directory (example):	description:
/usr/local/tipic/store	Location for static .htx and .t3i files

Although minimal configuration can be maintained without the following files, it is recommended that the user also downloads and updates the following:

file name:	description:	location
defstyle.css	Default Cascaded Style Sheet	/usr/local/tipic
addhtml.htx	Additional HTML template file	/usr/local/tipic
onpren.htx	Transaction End notification template	/usr/local/tipic/store
onerr.htx	Transaction error template	/usr/local/tipic/store

Please refer to Appendix B for a sample TIP Internet Connector installation procedure.

3.2. Configuring Apache Web Server for TIP/ic

Since TIP Internet Connector is a FastCGI application, most of the process management will be configurable through the Apache server FastCGI configuration. Apache has to be configured to run FastCGI applications, and should be able to find TIP Internet Connector and recognize it as a FastCGI application. This can be done either by dedicating a directory on a server to FastCGI applications, by registering TIP Internet Connector executable as FastCGI handler, or using an extension to achieve that.

Note: This last approach, based on registering FastCGI handler extensions, breaks the dynamic .htx template generation. TIP Internet Connector does not have an extension and links provided on dynamically created .hxt template files do not use one. However, this configuration works with static forms, provided that links in the .htx files are updated.

Other global FastCGI configuration directives could be used when controlling the behaviour of TIP Internet Connector. Please refer to the FastCGI documentation (e.g. www.fastcgi.com) for directives to control

the number of instances of TIP Internet Connector running, keeping one alive and so on.

A number of parameters specific to the TIP Internet Connector can be specified in its configuration file. The full path to this file can be supplied through an initial FastCGI environment variable, `TIPICCCONFPATH`. If not supplied, TIP Internet Connector will attempt to load its configuration from the `tipic.conf` file in `/usr/local/tipic/conf`.

More on Apache configuration for TIP Internet Connector and its installation in general is given in Appendix E.

3.3. Configuring TIP/ix Server for TIP/ic

TIP Internet Connector relies on TIP/ix support for DTP connections multiplexed over several sockets. The minimal version of TIP/ix that TIP Internet Connector can connect to as a TIP LOCAP is 2.3 R0 - 0069. Some additional tipix.conf parameters are also required for TIP Internet Connector.

PARAM WEBSESSIONS=n
: n is maximum web sessions allowed to be active. Is limited by the maximum users in the TIP/ix license

PARAM WEBALTER=(bgn,end,num,bgn,end,num)
: where bgn & end specify time of day in hh:mm format. If only two digits are given, they are assumed to specify the hh value and the mm value will be zero. For example 08 is 8:00AM, 17 is 5:00PM, etc… If the current time of day is between bgn & end, then WEBSESSIONS is altered to be n. You may supply 1 or 2 sets of alternate values.

PARAM WEBTIMEOUT=n
: n is minutes of idle time before timeout default is 120 minutes

PARAM WEBUNIXUSER=usr
: Unix User-id to be assigned to web users default is 'nobody'

PARAM WEBTIPUSER=usr
: TIP/ix User-id to be assigned to web users, default is 'WEBDFLT'

PARAM WEBPERFIX=x
: 1 or 2 letters used to prefix a unique terminal name. default is 'W'

LOCAP of the TIP/ic type should also be configured on TIP/ix, and the IP address of the machine running TIP Internet Connector should be supplied.

Web users connected to TIP/ix via TIP Internet Connector will be listed in whoson as term@locap in the display.

TIP/ix also requires that TIP/ic is registered on the TIP/ix system. For registration information please contact Inglenet.

3.4. Configuring TIP Internet Connector

A number of parameters specific to the TIP Internet Connector, and its default behaviour in particular, can be specified in a configuration file in a format somewhat similar to the Apache configuration file. The full path to this file is supplied through a FastCGI environment variable, as described in 3.2.

TIP Internet Connector configuration file maintains other two important blocks of information. TIP/ix system is referenced through a LOCAP. LOCAP is defined as a particular port on a particular host, thus allowing multiple TIP/ix systems running on the same UNIX box. Table of LOCAPs is loaded at the initialization time from the TIP Internet Connector configuration file.

Also, TIP Internet Connector can hide the real transaction name through an alias used in the link on the HTML source. In addition this alias can be used to specify that that particular transaction should always be executed on a requested TIP/ix system. The alias gets translated appropriately, based on the information stored in the same configuration file.

The TIP/ic configuration file may be edited manually, or viewed through the use of a TIP Internet Connector tool tipicconfig. tipicconfig is another FastCGI application that should be exposed through a link very similar to the tipic one:

```
http://www.mydomain.com/fcgi-bin/tipicconfig
```

3.5. Error Logging and Debugging

TIP Internet Connector logs information into its own log file. The level of information logging can be selected in the TIP Internet Connector configuration file. The logging routine used in TIP/ic is the same as the one used in the rest of TIP/ix. You defined the logging options as a string of values such as ""a3MK". A number before the letter M defines how many Megabytes of log information to keep before wrapping the log file around. The other possible options mean the following:

w	log warning messages
c	log commands
d	log more details
a	log warnings, commands and details
t	Trace level logging. Logs everything possible
C	dump TCP/IP packets sent to TIP/ix and received from TIP/ix
K	Keep the last 10 tipic.log files

3.6. Security

TIP Internet Connector does not provide for additional security; it relies on the one provided by the Apache server and TIP/ix system. However, it does hide some critical information in the HTML source code from the malicious user:

Session ID is always encrypted. Internally, three different session IDs are stored in the session table object shared between instances of TIP Internet Connector: two different session IDs are used in the DTP protocol, and they are mapped through the third one generated just for the HTML source. Session ID displayed in the browser is encrypted based on the local session id and the time/day stamp. This simple encryption was deemed sufficient to deter a malicious user.

Transaction ID can be mapped to an alias visible in the HTML code or URL

3.7. Internationalization

TIP/ix system can store data as 8-bit characters. In a standard, terminal emulation mode of use, 7-bit characters can be interpreted differently on different systems, based on the client locale configuration. Although viewed from TIP/ix as another passive TIP system, regarding internationalization TIP Internet Connector is no different than a single TIP/ix client.

Version 1 will not provide localization on a per user basis, but the entire system will use the same locale settings. Future versions of TIP Internet Connector can handle this through the Apache content negotiation to send the data to the browser in a language it can understand. This would require AddLanguage configured in Apache and different links for specific documents (one per language).

3.8 tipic.conf Parameters

This configuration file is in XML format and a sample with comments describing the parameters follows:

```
<TicConf>
<!--
  -- ServerRoot: The top of the directory tree under which
  -- the server's configuration, error, and log files are kept.
  -- Do NOT add a slash at the end of the directory path.
  -->
<ServerRoot value="/usr/local/tipic"/>

<!--
  -- ServerAdmin: Your address, where problems with the server
  -- should be e-mailed.  This address appears on some
  -- server-generated pages, such as error documents.
  -->
<ServerAdmin value="rjn@inglenet.com"/>

<LogFile value="/usr/local/tipic/logs/tipic.log" options="at3MK" />
<!--
  -- RegularStore: The location where converted .htx and .t3i
  -- files are regularly stored.
  -->
<RegularStore value="/usr/local/tipic/htx/store"/>

<!--
  -- Cache: The location where converted .htx and .t3i files
  -- are cached.
  -->
<Cache value="/usr/local/tipic/htx/cache"/>

<!--
  -- ThisLocap: The Locap for this machine and Internet Connector
  -- running on it (there could be multiple instances of the same
  -- locap, but there could also be multiple locaps, provided that
  -- we have multiple configuration files and multiple binaries:
  -- not very useful, but possible)
  -->
<ThisLocap value="POPAIC"/>
<!--
  -- AdditionalHtml: The file name of the text file contents
  -- of which is included at the bottom of automatically
  -- generated .htx file.
  -->
<AdditionalHtml value="/usr/local/tipic/htx/addhtml.htx"/>

<!--
  -- DefaultStylesheet: The file name of the .css file
  -- name of which is embedded as stylesheet file in the
  -- header of the automatically generated .htx file.
  -->
<DefaultStylesheet value="/tipic/defstyle.css"/>

<!--
  -- FccModsSupport: This options controls the complexity of the
  -- HTML templates created during the dynamic screen format
  -- conversion, depending on the required level of the support for
  -- the FCC Modifiers.  Permited values are full, limited and none
  -->
<FccModsSupport value="none"/>

<!--
  -- OnPRENPage: The file name of the .htx file that notifies
  -- the user that the transaction succeeded.
  -->
<OnPRENPage value="onpren.htx"/>
```

```
<!--
  -- OnErrorPage: The file name of the .htx file that notifies
  -- the user that the transaction failed, and gives more explanation.
  -->
<OnErrorPage value="onerr.htx"/>

<!--
  -- InstanceCleanup: the time-out for the cleaner process to clean
  -- individual instancess of TIP ic and session records, if not
  -- touched for a predefined time.
  -->
<InstanceCleanup value="180"/>

<!--
  -- SystemCleanup: time-out for the cleaner process to kill itself
  -- and release shared session table, if no process touches a
  -- table for a predefined amount of time.
  -->
<SystemCleanup value="900"/>

<!--
  -- MaxSessions: the maximum number of DTP sessions to all TIP
  -- locaps from this system.
  -->
<MaxSessions value="1000"/>
<!-- [Locap] element:
  -- Attributes:
  --     name - name of the Locap.
  -- Contents: [Address], [Port], optional [default] elements.
  --
  -- [Address] element: defines network address of the locap.
  --
  -- [Port] element: defines port of the locap for TIP/ix
  --
  -- [default] element: if present, makes this locap a
  -- default locap.
  -->
<Locap name="POPA" address="192.168.1.22" port="10555" />

<!-- [Transaction] element:
  -- Attributes:
  --     name - name of the transaction.
  -- Contents: [Id], optional [Locap] elements.
  -- [Id] element: defines id of the transaction.
  --
  -- [Locap] element: if present, defines name of the locap
  --     for this transaction.
  --
  -- [CDA] element: if present, defines the CDA area for this
  --     transaction, supplied when the trensaction is started.
  -- Attributes:
  --     value - complete CDA text for this transaction, spaces as '+' signs.
  -- [StyleSheet] element: if present, the name of the .css file
  -- which is embedded as stylesheet file in the header of the
  -- automatically generated .htx file, only the ones created
  -- during this transaction.
  -- Attributes:
  --     value - complete or relative path.
  -- [AddHtx] element: if present, the name of the AdditionalHtml file,
  -- contents of which is included at the bottom of automatically
  -- generated .htx file, only the ones created during this transaction.
  -- Attributes:
  --     value - complete or relative path.
  -->
<Transaction name="WEBTSP" id="TSP" locap="POPA" log="w"/>
```

```
<!---[AppGroup] Element:
  --- name : required and defines the Application group name
  --- URL  : required and defines the base part of the URL, This is most likely
  ---        defined using Alias in the Apache httpd.conf
  --- STORE : optional, defines the directory to search for .html, .htx & .t3i files
  --- LOG  : optional, defines the logging to be used when operating on the application
  --- HTML  : optional, if the initial input, then reply with the named html file
  --- TRANSACTION : optional, if the initial input, then run the defined tranasaction
              On Tip/ix
  --- CDA=  : optional, if TRANSACTION= is used, then this defines some data to
              Be placed in the CDA text area
 -->
<AppGroup name="TstApp"  URL="/txn/"  store="/usr/local/tipic/htx/store2" log="cw" />
<AppGroup name="TstApp2" URL="/txnx/" log="t" HTML="ron.html" />
<AppGroup name="TstApp3" URL="/txnt/" log="t" transaction=TSP CDA="TSPFILE " />

<!--[Setenv] Define environment variable values that can be referenced in the .htx files
 -->
<setenv name=”LocalHTTPS” value=”https://tasgcic.doas.state.ga.us” />
</TicConf>
```

4. TIP Internet Connector Templates

4.1. Introduction to HTML Templates and Data Definition Files

TIP Internet Connector request processing is based on MCS Screen Formats converted to HTML template files (.htx) and Data Defintion files (.t3i), containing COBOL oriented data description. Each MCS Screen Format is converted to a pair of file: one .htx file and one .t3i file. Conversion of MCS screen formats into .t3i and .htx files can be done at run-time, or through the use of a tool provided by Inglenet. Provided that the MCS Screen Format was successfully converted, TIP Internet Connector replaces place holders in the .htx files with data arriving from the TIP/ix system, honoring the data structure as provided in the .t3i file. The resulting dynamically created HTML page is presented to the end user.

In a typical scenario, TIP Internet Connector parses the initial query string (URL request) and other data coming from the Web server in FastCGI requests. Then it processes the request, invoking a transaction on a TIP/ix system. TIP/ix system responds with a data and a screen format name. Finally, a dynamically created HTML page is read and sent to the browser. With a page, an HTTP header is sent disallowing page caching, thus preventing users from browsing back and forth within and across TIP transactions. In the case of a failure, TIP/ic renders an appropriate error message, together with the error code. It also logs information in the Apache error log.

The request coming from the Web server gets parsed, detecting the presence of the ‘SID’ variable and activating that session appropriately. If no ‘SID’ was found, a new session is established before processing the request. Other keywords, such as ‘pib.transaction’, starting the TIP transaction and ‘mcs.name’ that persists the name of the MCS screen format (and the .htx/.t3i file pair) are honored. All these keywords together form the TIP Internet Connector’s object model.

All other variables are treated as fields on the form, with the exception of three reserved hidden fields used for MCS communication, McsCount, McsSize and McsStatus. These fields should not be changed manually, but rather through the JavaScripts provided in all .htx pages.

There are two special .htx pages that can be configured in the TIP/ic Configuration File: transaction notification templates (onpren.htx and onerr.htx by default). They are used to inform end users of successful TIP transaction termination, or erroneous condition, respectfully.

4.2. JavaScripts

All HTML templates converted from MCS Screen Formats contain several JavaScript functions at the beginning of the page, whiteout that they would not function correctly.

function DoXMIT(form, nCount, nRecord) is used instead of submit, to customize the amount of data supplied to the TIP/ix system. Some TIP applications might tailor they behaviour based on the amount of data they receive from the end user (i.e. detecting user's request based on the cursor position on the terminal screen). This can be emulated in an .htx file by altering the McsCount hidden field; typically this is done by supplying the amount of data that is required in the nCount parameter of the DoXMIT function:

```
<INPUT CLASS="FKEY" TYPE=BUTTON TITLE="Execute"
        VALUE="OK"
        onClick="DoXMIT(\{mcs.name}, 24, 1)">
```

function DoFKEY(form, nKey) is used to emulate the end user pressing a function key on the terminal. form is the HTML form name (usually \{mcs.name} variable is used), and nKey is the ordinal number of the function key, with '0' being Escape or Message Wait.

```
<INPUT CLASS="FKEY" TYPE=BUTTON TITLE="Next"
        VALUE="F2"
        onClick="DoFKEY(\{mcs.name}, 2)">
```

functions EnableFilter, DisableFilter and FilterInput are used to handle case conversion and character detection (necessary for numeric, alphabetic and uppercase fields) across browsers. The functions themselves can be improved to go beyond the rudimentary browser detection, should the need arise. The use of the functions will typically remain unchanged, from the onfocus and onblur events on INPUT fields.

4.3. Variable and Preprocessor Directives Notation

HTML template file is essentially an HTML page containing three types of information that gets dynamically expanded before the page is handed over to the end user via the Web server:

- MCS data variables
- TIP/ic variables (TIP/ic objects and properties, parts of its object model)
- TIP/ic HTML expressions (pre-processor directives)

4.3.1. Data Variables

Variable substitution in the HTML template file is indicated by the following placeholder:

```
\{variable}
```

The variable name is preceded by backslash and open brace and terminated by a closing brace. An exception to that are HTML expressions, where variables are not enclosed in the \{} string: within expressions variable is any attribute value or parameter in it that is not recognized as a string or numeric value. All variable names are case insensitive.

Data Variables are essentially individual fields from the original Screen Format. The format of the data variable (field) is stored in the matching .t3i file. The value of the data variable gets replaced at run time by the MCS data provided by the TIP/ix system, or the default value from the .t3i file (if such a value exists).

One or more Data Variables can be grouped. The layout of the group (or groups, which can be nested or sequential) is defined in the .t3i file. Groups are described in the section on TIP/ic object model.

One special variable \{SID} (and the matching keyword) are used to maintain session state information across requests. Session IDs are encrypted before being passed to the browser.

You may also use an valid expression as a variable for example:

```
\{(pib.yyyy - 1)}
\{substr(mydata,5,10)}
\{toupper(pib.month)}
```

4.3.2. TIP/ic Object Model

In addition to the textual data received from the TIP/ix system, .htx file can be extended with other types of information also received from TIP/ix (i.e. screen format name or terminal id) or state information coming from TIP/ic (i.e. session id or screen format sub-screen name). These parameters combined together in an object oriented manner represent the TIP/ic object model.

Object properties are specified in the .htx file in a manner similar to MCS Data Variables:

```
\{mcs.name}
```

Objects and properties are typically used within the HTML templates as placeholders. Properties are then 'read' and replaced with live data at run-time. Some of them can also be set, or used to supply some information relevant for the request. These properties are then 'written'.

E.g. the following initial link:

http://www.mydomain.com/tipic?pib.transaction=tsp&pib.locap=arc&cda.text=TSPFILE+

could be used to invoke TSP transaction on the TIP LOCAP ARC, passing 8 character CDA containing text "TSPFILE ". To achieve this, two objects and their three properties are set.

A simpler method of invoking a transaction would be using a link such as the following:

http://www.mydomain.com/txn/tsp

Providing the httpd.conf Apache config file has an Alias mapping /txn to tipic and the TIP/ic tipic.conf file defines an AppGroup for /txn/, the above link will start the 'tsp' transaction on the default LOCAP defined for TIP/ic.

In the object model each property is marked as read-only, write-only or read-write (R, W or RW respectively).

4.3.2.1. mcs Object

mcs object provides an interface to the information received from the TIP/ix system, other than the textual data. Its properties are:

name (RW)
Screen format name. Screen formats are assigned a name when they are defined using the TFD program. The format name may be up to eight characters in length and must start with a character that is not a digit.

terminal (R)
terminal id automatically assigned by the TIP/ix system. Terminal ids are automatically assigned on the TIP/ix system (please refer to the TIP/ix configuration section).

function (R)
specifies additional optional processing. Consult MCS subroutine descriptions in TIP Programming Reference for discussion of the relevant values of this field.

hold (R)
may be set to the value "L" on the TIPMSGO call to lock the terminal keyboard following delivery of the output message. This is useful only if multiple screens are received in succession, which is not supported by TIP/ic.

Therefore, by default this property does not affect TIP/ic behaviour.

size (RW)

the maximum number of bytes that is expected as a result of an input request. The TIP program can use this value to determine whether the data received on an input message represents a "full screen". This is discussed in the description of the TIPMSGI subroutine call in TIP Programming Reference.

status (RW)

this property indicates what type of terminal activity was detected: for example, MSG WAIT or a function key or XMIT. Various 88 level items are provided to simplify program coding, and are listed in TIP Programming Reference. This property gets set only on request and is therefore not used in most .htx files, with the exception of OnPren and OnErr (or similar transaction notification templates).

filler (R)

set to the desired "fill" character to use on output. Choices are: space, underscore or asterisk character. It could be used in pre-processor conditionals to set field protection (i.e. space would set field to protected or read-only, underscore to unprotected or input)

count (RW)

this field to contain a count of the number of data bytes in the MCS-DATA area that are output to the screen format. In transaction notification templates it could indicate the number of data characters sent. The input count is always less than or equal to the value that mcs reports in the size property (the maximum) and always includes the full size of the last field where the cursor was resting.

error (R)

error text as supplied on the TIPMSGE call. Please refer to TIP Programers Reference for more details.

fccMods (R)

fccMods object used in FCC Modifiers support.

page (W)

user can be lead to browse through several pages, collecting information that is too large to be displayed on a single HTML page, before submitting the request to TIP/ix. This is supported through the ‘mcs.page’ property that is set to the HTML subpage name in the form action attribute, or explicit links. All URLs that contain ‘mcs.page’ will not submit requests to TIP/ix. First next URL that does not

contain it would result in the TIP/ix request, submitting all the data collected in the meantime.

4.3.2.2. pib Object

pib object maps the TIP Process Information Block (PIB). TIP establishes a PIB area for each execution of a transaction program. pib object contains information about the transaction that is executing.

Its properties are:

transaction (RW)

TIP application that is invoked on the TIP/ix system.

Transaction name as specified with the 'pib.transaction' keyword can be no more than an alias that gets mapped to the real TIP transaction name at run time. For this purpose, TIP/ic internally maintains a loaded from the configuration file at the application initialization time, mapping transaction aliases to TIP transaction IDs and possibly LOCAPs. It could also contain CDA area and some screen format conversion parameters specific for that transaction. For more information please view comments in the default TIP/ic configuration file.

'pib.transaction' is always part of the initial URL.

user (R)

user id of the user that is executing the program on the TIP/ix system. User ids are not unique, as they are shared between all TIP/ic users at a given TIP locap (please refer to the TIP/ix configuration section)

terminal (R)

terminal id automatically assigned by the TIP/ix system. Terminal ids are automatically assigned on the TIP/ix system (please refer to the TIP/ix configuration section).

date (R)

current date that typically replaces a similar special field from MCS screen formats.

time (R)

current time of day that typically replaces a similar special field from MCS screen formats.

locap (RW)

contains the network name of the TIP/ix system where the program is running. TIP/ic resolves this name to a host name/port number pair to be able to connect to the correct

system. The table of LOCAPs and the default LOCAP are supplied in the TIP/ic Configuration File.

'pib.locap' is an optional par of the initial URL; if not supplied default LOCAP is used.

yyyy current 4 digit year

yy current 2 digit year

mm current 2 digit month

dd current 2 digit day of month

hh current 2 digit hour of the day (24 hours)

min current 2 digit minute of the hour

sec current 2 digit seconds of the minute

month current Month of the year as character string

day current Day of the week as character string

4.3.2.3. err Object

err object is valid only on the unsuccessful transaction termination template (onerr.htx or similar, depending on the system configuration). It has only one property and numerous constants that can be used in pre-processor conditionals:

code (R)
property that contains the failure code

Code	Description	Value
Unknown	there was some error but none of the more specific error codes apply	-1
FileNotFound	operation relied on the existence of the file, which was not found	-2
NotImplemented	operation not implemented in this version of the software	-3

Code	Description	Value
NoMemory	not enough memory could be allocated	-4
NoResource	not enough system resources (file handles, sockets) to complete the operation	-5
BadFormat	bad file format. Typically, this means a syntax error	-6
InvalidParam	bad or unsupported parameter was passed	-7
AppGeneric	there was some error in core objects, but none of the more specific error codes apply	-100
CommTransTerminated	transaction was successfully terminated	1001
CommUnexpectedAccept	error in communication with TIP/ix, session requests and replies are out of sync	1002
CommGeneric	there was some error in the communication with the TIP/ix system, but none of the more specific error codes apply	-1000
CommDisconnected	session expired and it was terminated by the cleaner process.	-1001
CommKeepAlive	attempt to set the keepAlive socket option failed	-1002

Code	Description	Value
CommReadLen	read attempt on the socket failed	-1003
CommBadDtpReply	unexpected reply came back from the host on the DTP message, the host is not the one supporting the DTP protocol, as we expected, or some other error protocol related error occurred on the host.	-1004
CommNoHost	TIP/ix host name could not have been resolved to a valid IP address	-1005
CommBadLocapTip	TIP/ix LOCAP was not found in the internal table, or is invalid for other reasons	-1006
CommBadLocapIc	invalid LOCAP supplied in the configuration file for TIP/ic	-1007
CommTipNotActive	the remote TIP system was not active	-1008
CommClosed	TIP/ix server has not been started. Or it might have been restarted and aborted your session.	-1009
CommScreenFormatNotFound	screen format that was requested does not exist on the remote system	-1010
FormatGeneric	MCS screen format conversion failed,	-2000

Code	Description	Value
	there was some error in the but none of the more specific error codes apply	
FormatUnknownError	same as FormatGeneric	-2001
FormatFileReadError	file read error during MCS screen format conversion	-2002
FormatConversionError	MCS screen format conversion itself failed, most likely corrupted screen format	-2003
ParserGeneric		-3000
ParserFileNotFound	file that was expected to be parsed (.htx, .t3i or .conf) does not exist	-3001
ParserBadDataRecord	data record not initialized or incorrect	-3002
ParserBadMcsData	internal parser error, MCS data buffer not initialized or incorrect	-3003
ParserInvalidCachePath	HTML template and data definition cache path as configured in the TIP/ic configuration file does not exist, or the Apache user does not have sufficient access rights	-3004
ParserInvalidRegularStorePath	HTML template and data definition permanent store path as configured in the TIP/ic configuration file	-3005

Code	Description	Value
	does not exist, or the Apache user does not have sufficient access rights	
SessTblRecordLocked	warning: an attempt was made to access shared session information on a session that was locked by another process	4002
SessTblIPCError	Inter Process Communication error, failed to access shared session information but none of the more specific codes apply	-4000
SessTblLockError	Inter Process Communication error, failed to access shared session information	-4001
SessTblLockFileError	failed to lock the file with the shared session information (but not the error of not locking file due to other processes locks). Most likely due to access rights.	-4002
SessTblShmAttachError	Inter Process Communication error, failed to attach the existing shared memory block	-4003
SessTblFull	exceeded the maximum number of sessions specified in the TIP/ic configuration file, no more space in the	-4004

Code	Description	Value
	session table,	
SessTblInvalidSessId	invalid session id was passed as a parameter, session probably expired in the meantime.	-4005
SessTblLatestMcsDataDir	can't create latest mcs data directory, usually access rights	-4007
SessTblLatestMcsDataFile	can't open, read or write to latest mcs data file; usually access rights	-4008
SessTblInvalidSessGuid	invalid session id was passed as a parameter, session probably expired in the meantime	-4009
TipAborted	TIP transaction aborted	-5001
TipEof	TIP end of file reached (pib status PIB-EOF)	-5002
TipIo	TIP I/O error (pib status PIB-IO-ERROR)	-5003
TipSecurity	TIP security error (pib status PIB-SECURITY)	-5004
TipNotFound	TIP not found error (pib status PIB-NOT-FOUND)	-5005
TipTimeout	TIP transaction timed-out (pib status PIB-TIMED-OUT)	-5006
TipNoMem	not enough resources on the TIP/ix side, including exceeding the	-5007

Code	Description	Value
	number of Web sessions configured in the TIP/ix configuration file (pib status PIB-NO-MEM)	

4.3.2.4. cda Object

cda object provides access to the TIP CDA area. This object can be used only for supplying information via CDA on the initial link, passing the CDA information when starting the application, or from the transaction termination notice template (onerr.htx), when the transaction terminates and CDA is returned.

text (RW)

currently text is the only supported cda property, containing the entire CDA (e.g. cda.text=TSPFILE+++++++++ would supply a 16 character CDA area containing text "TSPFILE")

4.3.2.5. [group] Object

group is a collection of fields grouped together in the .t3i file for the benefit of simplifying HTML template through the use of loops. It is almost exclusively used within a pre-processor loop, to control the amount of information that gets generated. It has the following properties:

max (R)

This property is created for each .t3i group (at runtime, this variable would be created for each named #BEGIN loop, other than when BEGIN INLINE="YES"). It is set to the maximum number of occurrences that the loop could be processed, as per the matching 'occurs' value from the .t3i file.

index (R)

index is updated for each iteration of the #BEGIN loop. This would hold values of 1, 2, 3, etc. up to 'count'. After the loop, the variable is removed. Before entering the block, the value is set to zero.

count (R)

this property reflects the number of occurrences that the loop will actually be processed, the number of non-empty records up to the 'max' value.

4.4. TIP/ic HTML expressions

TIP/ic HTML expressions are essentially pre-processor directives in the template, conditionals, loops or similar, that generate the HTML template before it gets handed to the variable processor.

TIP/ic implements HTML expressions in a manner similar to XFCGI. Currently, that includes #if, #else, #endif based mostly on the syntax defined for '.shmtl' files (Server Side Includes). It also supports #begin, #end loops. Each directive is actually enclosed in an HTML comment. They start with <!-- and end with -->. This allows the HTML template file to be easily edited with any HTML text editor.

The '#' character must be the first character after the <!-- otherwise the XFCGI & TIP/ic HTML parser will also assume the sequence to be a comment. There may also be spaces after the '#'. So each of the following would be legal:

```
<!--#endif-->
<!--# endif-->
```

XFCGI & TIP/ic will process the directive up to the trailing '-->' or the first colon (':'):

```
<!--#if expr="variable gt 'Fred'" : And now for
some comments -->
```

When processed, none of the directive itself would be sent to the web browsers, only the results of the processing. I.e. we do not want to see "`<!--#if stuff -->`" appearing at the browser.

Variables that are used within HTML expressions are not enclosed in the \{} string.

4.4.1. Conditionals

Conditional blocks in TIP/ic are supported through the #if, ,#else-if, #else and #endif tags.

#if tag has a single attribute, expr. Variables are replaced within the value of expr, before the expression gets evaluated. This can result in one of two logical states: true or false. Based on that the conditional block either gets included into the HTML template, before its processing, or it gets excluded from it.

#else and #endif tags do not take any attributes. #if conditionals can be complex or nested:

```
<!--#if expr="(mcs.filler NE ' ' AND mcs.filler
NE '_')
```

```
                OR field_007 GT ' '"-->
        firstIf
            <!--#if expr =  " mcs.filler LT '_'"-->
                scndIf
            <!--#else-->
        scndElse
            <!--#endif-->
<!--#else-if expr="myfld eq 'Bob'"-->
            Else if condition do this
<!--#else-->
        firstElse
<!--#endif-->
```

4.4.2. Loops

HTML templates can be simplified through the use of loops. Loops are delimited with the #begin and #end tags. Loops should always be named.

#begin tag has two attributes: name and max. name is a string attribute that provides a name of the loop. This name should match the name of the group in the Data Definition file (.t3i). max is the number of times that the loop will be processed. It could be set to a numeric value, or one of the group properties (such as 'group.count' or 'group.max').

Loops can also be nested.

```
<!--# begin name=subitems max=subitems.max -->
<TD>
  <SPAN LASS="DATA_READONLY">\{CustNum1}</SPAN>
  <INPUT TYPE=HIDDEN NAME="CustNum1"
VALUE="\{CustNum1}">
  <SPAN LASS="DATA_READONLY">\{CustNum2}</SPAN>
 <INPUT TYPE=HIDDEN NAME="CustNum2"
VALUE="\{CustNum2}">
</TD>
<!--#end-->
```

Numeric or String

When doing computes and/or comparisons, the value of a variable has the trailing spaces removed. But if the value was all spaces, then it is handled like it was just one space. Leaving one space allows us to distinguish between all spaces and empty.

Expression syntax

Parenthesis may be used for the usual meaning.

Numbers are coded as just numbers.

Strings are enclosed in single quotes.

All arithmetic operations are integer type.

The following arithmetic operators (using integer arithmetic) are to be supported:

+	Addition
-	Subtraction
/	Division
*	Multiplication
%	Remainder

The following **numeric** comparison operators are supported.

EQ	Equal to
NE	Not equal to
GT	Greater than
LT	Less than
GE	Greater than or equal
LE	Less than or equal

The following **string** comparison operators (case insensitive) are supported.

==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
Contains	Does op1 contain op2 with in it?
Begins	Does op1 begin with the string op2?

Boolean operators

AND	Logical AND
&&	Logical AND
OR	Logical OR
||	Logical OR

Unary operators

!	Logical NOT
-	Arithmetic minus

Assignment operator

=	Assign, eg. Wrkn = 3 * 5

Some examples:

```
<!--#IF expr="((blk1.index / 3) % 2 eq 0)" : If even group of 3 -->
<!--#IF expr="blk1.index eq '3'" : If third occurrence -->
```

One variation is to simple test if a variable has any data at all.

```
<!--#IF expr="discPrice ne ''"-->
```

If `discPrice` is present and has a value other than a null string (`''`), then the HTML following the #IF is processed.

When the CONTAINS operator finds a match it will set the value of `contains.index` to the position within the string where the match happened. An example of using CONTAINS follows:

```
<!--#IF expr="HTTP_USER_AGENT contains 'MSIE'" -->
 MS IE found at position <!--#echo var="contains.index" -->
 <!--#ELSE -->
 Likely using Netscape
 <!--#ENDIF -->

<!--#set var="wrkval" compute="3 * (wrkv2 = wrkv3 = 4 * 5)" -->
```

Functions

A few simple functions have been implemented. The function names are case insensitive, so Max is the same as max is the same as mAx, etc. For example max(4,10*2,7) would return 20. The following table lists the functions implemented.

Function	Min # params	Max # params	**Description**
min	2	20	Computes the minimum numeric value
max	2	20	Computes the maximum numeric value
sum	2	20	Compute the sum of all parameters
avg	2	20	Compute the average of all parameters
abs	1	1	Compute the absolute value of the parameter
toupper	1	1	Convert the string to upper case
tolower	1	1	Convert the string to lower case
length	1	1	Compute the length of the string for the parameter
substr	2	3	Return the substring of parameter 1. Substr(field,pos,len) starts at position pos for length len Substr(field,pos) starts at position pos to end of field.
concat	2	20	The string value of the parameters are concatenated left to right.
printf	2	20	Simple printf function. The 1st parameter is the format string. Only the simple cases of %s and %d are recognized. Parameters 2 through n are merged into the format string according to the positions of %s & %d and the resulting string is the value processed next.
soundex	1	1	The soundex value as defined by Knuth for the string parameter is computed.

Now some examples:

```
<!--#set var="wrkval" compute="Min(2*3,4,5*6) * Max(-12,5*8,12/2,3,5)" -->
 For the values -12,5*8,12/2,3,5,200,
 Min <!--#echo var="wrkval" compute="Min(-12,5*8,12/2,3,5,200)" -->,
 Max <!--#echo var="wrkval" compute="Max(-12,5*8,12/2,3,5,200)" -->,
 Avg <!--#echo var="wrkval" compute="Avg(-12,5*8,12/2,3,5,200)" -->,
 Sum <!--#echo var="wrkval" compute="Sum(-12,5*8,12/2,3,5,200)" -->. </p>
 <p><!--#echo var="testcat" value="Test on \{pib.baseUrl} received via
\{HTTP_REFERER}" -->
 <!--#echo var="wrkval" compute="-((3 + 5) * 6)" -->
 <!--#echo var="wrkval" compute="tolower('FooBar')" -->
 <!--#echo var="wrkval" compute="toupper('FooBar')" -->
 <!--#IF expr="HTTP_USER_AGENT contains 'MSIE'" -->
 <p>Using MS IE found at position <!--#echo var="contains.index" -->
 of <!--#echo var="HTTP_USER_AGENT" --> </p>
 <p>Upper=<!--#echo var="wrkval" compute="toupper(HTTP_USER_AGENT)" -->
 This is <!--#echo var="wrkval" compute="length(HTTP_USER_AGENT)" --> bytes long.
<p>Test concat '<!--#echo var="wrkval"
 compute="concat('IE version is ',
                 substr(HTTP_USER_AGENT, contains.index, length(HTTP_USER_AGENT) -
                        contains.index),
                 ', that was fun!')" -->'
 <!--#ELSE -->
 Likely using Netscape \{printf(' in the year %04d',pib.yyyy)}
 <!--#ENDIF -->
```

Appendix A

Technical Discussion on Major Objects

Several major functional components can be identified in the TIP Internet Connector.

Core component, providing integration with the Web server and gluing other components together. This component also handles configuration.

Communication component, maintaining connections to the TIP/ix system and handling the DTP and network protocols. This component maintains session state.

HTTP Conversion engine, handling the DTP to HTTP and HTTP to DTP data conversions.

MCS Screen Format parser, converting MCS screen formats at run time to the HTML template and data definition.

Components are organized in several static libraries (libticcommon.a, libticparser.a and libmcsformat.a) and one dynamic (libexpat.so). DTP objects and core object are maintained in the tipic binary.

The following diagram gives an overview of major TIP Internet Connector components and their relationships.

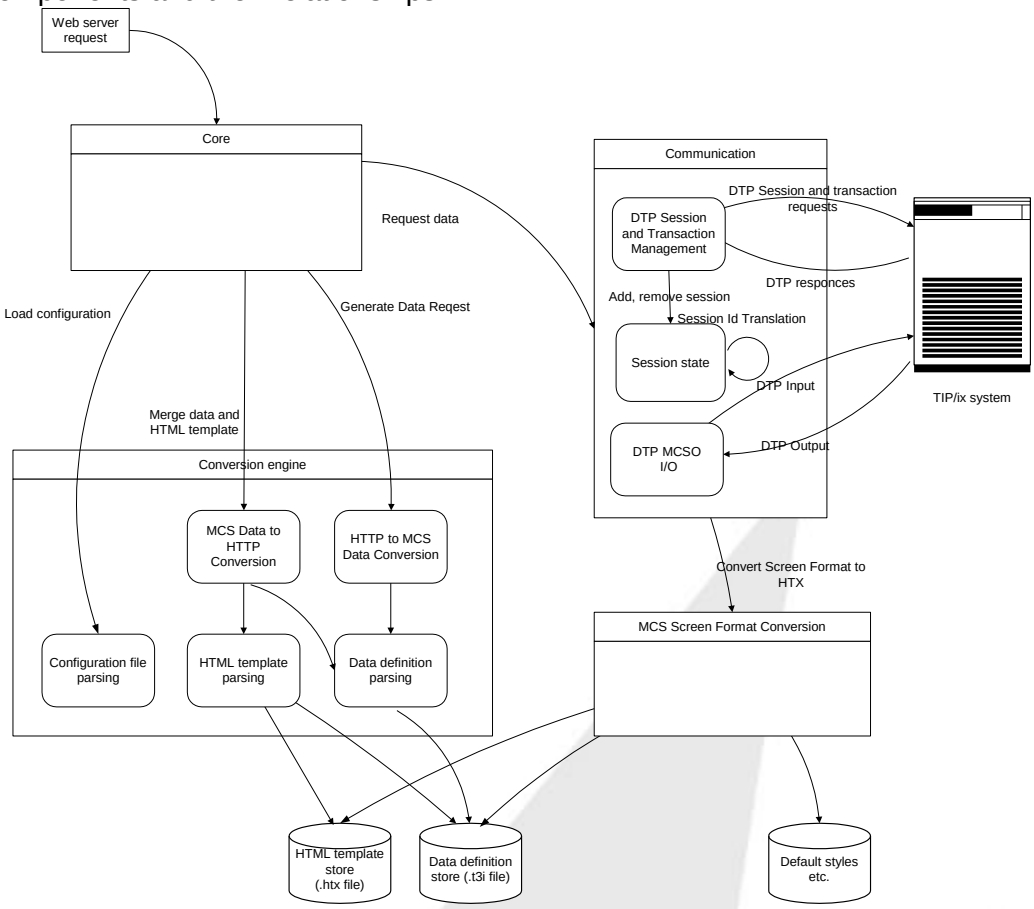


Fig. 2. Major subcomponents of the TIP Internet Connector

The following section gives an insight into the TIP Internet Connector design, describing major objects and the functionality they provide. This discussion is more technical than the rest of the document and it is not necessary for the successful installation and operation of the TIP Internet Connector.

However, the following technical details might prove valuable when troubleshooting problems in building, installing or running TIP Internet Connector.

A.1 Core and Communication Objects

Communication objects use DTP protocol, working with sockets in the blocking mode – each DTP MCSO input message being sent should result in the response read on the same socket. The response can be a MCSO output message (with the screen format name and data), one that gets translated into the HTML data and returns control to the browser. Or

it can be the DTP program end message (PREN) for this or another session. The former happens when program normally terminates or aborts without any output. The latter happens when TIP/ix seizes an opportunity of the existing conversation to notify TIP Internet Connector of a pending PREN for a session that either terminated with an output (so MCSO was received, but PREN could not due to the synchronous conversation mode), or aborted since sending the response. In this case DTP Protocol will clear the session entry in the table and keep reading the response to MCSO it originally expected.

Shared session table maintains session information shared between instances of TIP Internet Connector, holding session ids and states for all TIP Internet Connector DTP sessions on the system. TIP Internet Connector allows multiple connections to the same TIP/ix LOCAP, and can run in multiple instances (invoked by the Apache through Fast CGI). If one instance of TIP Internet Connector is blocked processing a request, a new one will be created. This means that the same DTP session can exist in multiple instances

Example:

An instance of TIP Internet Connector creates session 0004 to locap ARC, but then becomes busy processing another session. Next request for session 0004 will result in a new instance of TIP Internet Connector connecting to ARC, thus having effectively two instances maintaining the same session. Furthermore, PREN for session 0004 can be received on any connection to ARC from any instance of TIP Internet Connector on this system, since they are seen as a single LOCAP from the TIP/ix DTP (the first instance expecting any MCSO reply will receive the pending PREN). Receiving this semi-asynchronous PREN would remove the session entry from the table, and all instances trying to access the same session later will discover that the session was terminated and clean their state.

Shared session table lives in a shared memory. Each session has one fixed size entry in the table. The presence of the shared memory block can be detected using the UNIX ipcs command.

Not all session state information is kept in the shared memory. Every session keeps its last MCS data in the Shared Session Table, so that it can retrieve it and merge on MSGE calls with high values. Last MCS data is not kept in the shared memory, but in individual files named by the session's encrypted id. These files get removed from the system as the session entry gets removed from the Shared Session Table. Look for the latestmcsdata subdirectory on your system, to find these temporary files.

When TIP Internet Connector detects it is the first instance running, and it does so through the Shared Session Table object, it forks a second instance of itself, a cleaner process. Cleaner process occasionaly wakes up (every one fifth of the Instance Cleanup period, as configured in the configuration file) and invokes instance timeout check. If an instance

expires, it gets killed and cleaner process sends the CLOSE message to the LOCAP that instance was connected to.

When there are no instances using the shared memory through the Shared Session Table, other than the Cleaner process itself, the shared memory will time-out and be released. This is controlled through the SystemCleanup entry in the configuration file.

Note: Under some Apache configurations (e.g. when it keeps the last instance of FastCGI application running), this will never happen.

TIP Internet Connector objects report errors using a global Results enumerator. All result codes are listed in with the TIP/ic err object. These error codes can appear on some error pages or in the error log file.

A.2. HTTP Conversion Objects

The expat parser is used to build a parsed tree of the HTML elements or data definition file.

The same parser is used to read the TIP Internet Connector configuration file. It also loads two maps: transaction alias to transaction name and info, and LOCAP name to host and port number. All this information is loaded during the application initialization and maintained for the life of the instance. For configuration changes to take effect it is necessary to kill the existing instance of TIP Internet Connector and allow Apache to create a new instance.

A.3. MCS Screen Format Conversion Objects

TIP Internet Connector merges the MCS data it receives from the TIP/ix system with the HTML template stored in the .htx file, using additional files for data description (.t3i), cascaded style sheet URL and default template extensions (.htx more likely than .html). HTTP Conversion objects rely on the existence of most of these files at appropriate locations.

MCS converter is used to convert an MCS format pulled from the TIP/ix in the case if pair of .htx/t3i files corresponding to the format name does not exist.

The file the contents of which will be inserted into the generated HTML immediately before closing </FORM> tag is configurable. This is useful for integrating a standard set of buttons into all dynamically created pages.

Appendix B

Installing and Configuring TIP/ic

October 29, 2010

t is our intention to provide a standard installation for TIP/ic similar to installing TIP/ix. All the files necessary for tipic installation are provided in a single tar file, tipicPLATFORM.tar, where PLATFORM is one of AIX, REDHAT3 or REDHAT5.

tipicPLATFOMR.tar can be downloaded from the Inglenet Web site or directly from ftp://ftp.inglenet.com/Products/TIPic/tipicPLATFORM.tar

1) download tipicPLATFORM.tar (where PLATFORM is one of AIX, REDHAT3 or REDHAT5). Extract the contents of the tar file in a working directory (like $HOME/mytmp) and then as root (or via sudo) run the 'install' script. The 'install' script will place the TIP/ic software and sample configuration files into the /usr/local/tipic directory. If you already have TIP/ic installed in /usr/local then you may want to take a backup copy of what you currently have or at least save your current tipic.conf file. The install will insert a sample tipic.conf file.
2) You should end up with the following files:

```
./conf:
tipic.conf

./docs:
tipic_manual.html

./bin:
tipic
tipicconfig

./htx:
addhtml.htx
defstyle.css

./htx/tipic/cache:

./htx/tipic/store:
onerr.htx
onpren.htx
```

Depending on the Apache user configuration, it might be necessary to explicitly set the library path with other initial environment variables passed to FastCGI applications. To do that you would need to add the following line to the Apache configuration file:

```
FastCgiConfig -initial-env
LD_LIBRARY_PATH=/usr/local/lib:/usr/lib:/usr/local/tipic/lib:
```

beware that only the last FastCgiConfig (of possibly multiple ones) is honoured. So if your configuration requires multiple FastCGI configuration directives, they should all be passed in a a single statement. So your FastCgiConfig directive is more likely to look like the following one:

```
FastCgiConfig -appConnTimeout 300 \
-initial-env TIPICCONFPATH=/usr/local/tipic/tipic.conf \
-initial-env LD_LIBRARY_PATH=/usr/local/lib:/usr/lib:/usr/local/tipic/lib
```

FastCGI enable Apache server. Please refer to the Apache Configuration section in this manual for more information on enabling Apache for FastCGI and registering tipic as a FastCGI application.

Also, make sure that it passes all the environment variables necessary for correct operation of the TIP/ic server (see step 2 for configuring complex FastCGI statements):

FastCgiConfig –initial-env TIPICCONFPATH=/usr/local/tipic/tipic.conf

move tipic_manual.html (this file) to your Document Root for future reference.

at this point you should be able to run tipicconfig from the Web Browser (http://mywwwserver/fcgi-bin/tipicconfig). Make system specific changes to the tipic configuration file. This includes all locations, but most importantly TIP/ic Locap (or ThisLocap) and all TIP LOCAP names, host names and ports. ThisLocap is the name of the TIP/ic system on the TIP network. LOCAPS are TIP names for all TIP/ix systems that are required to be accessed from this TIP/ic. Please refer to the section of this manual on TIP/ix configuration for more details.

If using fcgid the example is:

```
# FastCGI
FcgidIPCDir /tmp

FcgidInitialEnv  TIPICCONFPATH   /usr/local/tipic/conf/tipic.conf
FcgidInitialEnv  LD_LIBRARY_PATH /usr/lib:/usr/local/lib:/usr/local/tipic/lib
FcgidInitialEnv  LIBPATH         /usr/lib:/usr/local/lib:/usr/local/tipic/lib
FcgidInitialEnv  PATH           /usr/local/bin:/usr/local/tipic/bin:/usr/bin:/bin
```

You might also use:

```
ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"
ScriptAlias /fcgi-bin/ "/usr/local/tipic/bin/"
Alias /txn "/usr/local/tipic/bin/tipic"
Alias /txnt "/usr/local/tipic/bin/tipic"
```

Appendix C

Data Definition File (.t3i) Format Example

Data definition files are generated by the MCS Screen Format Converter (or convt3z). Typically they do not require user intervention. In this they differ from their peer files, HTML templates (.htx).

However, they are cases when user might find it advantageous to customize data definition files: most likely scenario is to simplify HTML templates through the use of #begin loops in templates and groups in data definition files. For such cases it is beneficial to understand the format of the t3i file.

Data definition files are stored in the following XML styled format (the following file was automatically generated by TIP Internet Connector MCS Screen Format Conversion component from the TF$TSP1A screen format):

```
<record>
    <field name="field_007" mask="U(2)"/>
    <field name="field_010" mask="U(3)"/>
    <field name="field_012" mask="9(5)"/>
    <field name="updatedby" mask="X()"/>
</record>
```

where the entries are structured according to the following description:

Root:

- Contents

one or more <record> elements (corresponding to Mcs data)

Element <record>:

- Contents:

one or more <field>

one or more <group>

Element <group>:

- Contents:

Atrributes 'name' and 'occurs'

one or more <field>

Attribute 'name': name of a group.

Contents:

an identifier (string of letters, digits and underscores starting from letter or underscore).

Attribute 'occurs': number of occurrences of the group.

Contents: number

Element <field>: description of a field .

Contents :

Attributes 'name', 'mask' or ‘default’.

Attribute name: name of a field.

Contents:

an identifier (string of letters, digits and underscores starting from letter or underscore).

Attribute mask: mask for the field.

Contents:

valid field mask as defined by MCS format standard.

Attribute default: default value of the field.

Contents:

textual data used as the value of the field when MCS low values supplied from the TIP/ix host, as defined by MCS format standard.

Appendix D

HTML Template File (.htx) Format Example

The following template file is based on the automatically generated from the TF$TSP1A MCS screen format, at run time. The last part of the file, follwing the "<!-- Following is a contents of the file addhtml.htx -->" comment line is essentially a content of the additional htx file automatically appended to each HTX file (files on their own, without any code to submit the page, cannot be used. For dynamically created pages it is necessary to add some default buttons or links).

```
<HTML>
 <HEAD>
  <SCRIPT LANGUAGE=JAVASCRIPT>
   var gFilter = false;
   var gToUpper, gAlpha, gNumeric;

   function DoFKEY(form, nKey) {
    if (nKey >= 0 && nKey <= 22) {
     form.McsStatus.value = "0123456789ABCDEFGHIJKLM".substr(nKey, 1);
     form.submit();
    }
   }

   function DoXMIT(form, nCount, nRecord) {
    if (nRecord <= 0)
     form.McsCount.value = nCount;
    else
     form.McsCount.value = nCount + (nRecord - 1) * 0;
    form.submit();
   }

   function EnableFilter(toupper, alpha, numeric) {
    gToUpper = toupper;
    gAlpha = alpha;
    gNumeric = numeric;
    gFilter = true;
   }

   function DisableFilter(fld) {
    if (document.layers && gToUpper)
     fld.value = fld.value.toUpperCase();
    gFilter = false;
   }
```

```
function FilterInput(e) {
    if (gFilter) {
     var ch;
     if (document.layers)
      ch = String.fromCharCode(e.which);
     else
      ch = String.fromCharCode(event.keyCode);
     if (gAlpha && ch >= ' '
         && (ch < 'a' || ch > 'z')
         && (ch < 'A' || ch > 'Z')) {
      return false;
     }

     if (gNumeric && ch >= ' '
         && (ch < '0' || ch > '9')
         && ch != '-'
         && ch != '$'
         && ch != '.'
         && ch != ',') {
      return false;
     }

     if (gToUpper) {
      if (!document.layers)
       event.keyCode = ch.toUpperCase().charCodeAt(0);
     }

     return true;
    }
    return true;
   }

   if (document.layers)
    document.captureEvents(Event.KEYPRESS);
   document.onkeypress = FilterInput
  </SCRIPT>
  <TITLE></TITLE>
   <LINK REL="stylesheet" HREF="/tipic/defstyle.css" TYPE="text/css">
 </HEAD>

 <BODY>
  <FORM NAME="\{mcs.name}" METHOD=POST
        ACTION="tipic?sid=\{sid}&amp;pib.transaction=\{pib.transaction}
        &amp;pib.locap=\{pib.locap}&amp;mcs.name=\{mcs.name}
        &amp;mcs.filler=\{mcs.filler}">
   <TABLE>
    <COL WIDTH="10%">
    <COL WIDTH="71%">
    <COL WIDTH="11%">
    <COL WIDTH="1%">
    <COL WIDTH="7%">
    <TD>
     <SPAN CLASS="SPECIAL">\{mcs.name}</SPAN>
    </TD>
    <TD>
     <SPAN CLASS="SPECIAL">\{pib.date}</SPAN>
    </TD>
    <TD>
     <SPAN CLASS="SPECIAL">\{pib.time}</SPAN>
    </TD>
   </TABLE>
   <TABLE>
    <COL WIDTH="15%">
    <COL WIDTH="85%">
    <TD>
     <SPAN CLASS="HEADING">TIP/ix Sample Program</SPAN>
    </TD>
    </TABLE>
    <TABLE>
     <COL WIDTH="7%">
     <COL WIDTH="44%">
```

```
<COL WIDTH="2%">
<COL WIDTH="2%">
<COL WIDTH="2%">
<COL WIDTH="43%">
<TD>
 <SPAN CLASS="HEADING">Enter the function to be performed:</SPAN>
</TD>
<TD>
 <INPUT CLASS="DATA_NORMAL" TYPE=TEXT
        onfocus="javascript:EnableFilter(true,false,false)"
        onblur="javascript:DisableFilter(this)" NAME="field_007"
        SIZE=2 MAXLENGTH=2 VALUE="\{field_007}">
</TD>
<TD>
 <SPAN CLASS="ERROR">\{mcs.error}</SPAN>
</TD>
<TD>
 <SPAN CLASS="HEADING">Enter customer key:</SPAN>
</TD>
<TD>
 <INPUT CLASS="DATA_NORMAL" TYPE=TEXT
        onfocus="javascript:EnableFilter(true,false,false)"
        onblur="javascript:DisableFilter(this)" NAME="field_010"
        SIZE=3 MAXLENGTH=3 VALUE="\{field_010}">
</TD>
<TD>
 <SPAN CLASS="HEADING">-</SPAN>
</TD>
<TD>
 <INPUT CLASS="DATA_NORMAL" TYPE=TEXT
        onfocus="javascript:EnableFilter(false,false,true)"
        onblur="javascript:DisableFilter(this)" NAME="field_012"
        SIZE=5 MAXLENGTH=5 VALUE="\{field_012}">
</TD>
</TABLE>
```

```
    <!-- Following is a contents of the file
         /usr/local/apssl/htdocsTest/tipic/addhtml.htx -->
    <TABLE WIDTH="100%" CLASS="FKEY_TABLE">
     <TR>
      <TD>
       <INPUT CLASS="FKEY" TYPE=BUTTON TITLE="Execute" VALUE="OK"
              onClick="\{mcs.name}.submit()">
       <INPUT CLASS="FKEY" TYPE=BUTTON TITLE="End" VALUE="Cancel"
              onClick="DoFKEY(\{mcs.name}, 0)">
       <INPUT CLASS="FKEY" TYPE=BUTTON TITLE="Next" VALUE="F2"
              onClick="DoFKEY(\{mcs.name}, 2)">
      </TD>
     </TR>
    </TABLE>
    <!-- End of a  contents of the file
         /usr/local/apssl/htdocsTest/tipic/addhtml.htx -->
   </FORM>

  </BODY>
</HTML>
```

Appendix E

This appendix lists some common steps necessary for configuring Apache server for FastCGI. Additional entries could be necessary, such as FastCGI expiry time.

E.1. Installing Apache on Red Hat Linux

Please refer to the Red Hat Linux documentation for installing Red Hat Linux and Secure Server. Red Hat Linux comes with a version of Apache WWW Server (Secure Server) that supports FastCGI applications.

When installing binary distribution of the Apache server (such as Secure Server), it is necessary to load modules in addition to the ones provided with the product, using Apache DSO functionality. At the very least mod_fastcgi.so has to be added. Therefore, in addition to the three packages that are required for the Secure Server installation (apache, mod_ssl and openssl), apache-devel package containing include files, header files and the APXS utility is also required.

Install Apache

The following example illustrates Secure Server installation following the Red Hat 7.0 Workstation insallation :

```
mount /mnt/cdrom
cd /mnt/cdrom/RedHat/RPMS
rpm -Uvh apache-2.2.*
rpm -Uvh mod_ssl*
rpm -Uvh openssh*
rpm -Uvh apache-devel*
rpm -Uvh apache-manual*
cd
umount /mnt/cdrom
eject /mnt/cdrom
```

Install FastCGI

To start mod_fastcgi.so installation, the following file is required:

mod_fastcgi_2.2.8.tar (www.fastcgi.com)

To unpack this archive and install it on the system, run the following tar command. After that refer to the mod_fastcgi installation documentation (an example is given here):

```
tar -xzf mod_fastcgi*
cd [modfcgi directory]
apxs -o mod_fastcgi.so -c *.c
apxs -i -a -n fastcgi mod_fastcgi.so
```

Configure Apache

Update the httpd.conf (or httpsd.conf) file in the **/etc/httpd/conf** directory for FastCGI

After this run the htpasswd command to generate basic authentication file, used during the Apache configuration for tipicconfig. Check man pages for more information on the utility. Other types of authentication are also possible.

The following listing highlights the lines that need to be added or changed in the file. The highlighted lines have a comment attached to them which explains what the change is for. By installing mod_fastcgi.so LoadModule and AddHandler entries were automatically added to the configuration file. Check these new entries and make sure that they are pointing to the correct location (e.g. /modules/mod_fastcgi.so). Typically, authentication is required for the tipicconfig tool. The following Apache configuration file includes entries required for that, please refer to the Apache documentation for the tools necessary to generate user name and password entries on the system.

```
    ...
#
# ScriptAlias: This controls which directories
contain server scripts.
# ScriptAliases are essentially the same as
Aliases, except that
# documents in the realname directory are
treated as applications and
# run by the server when requested rather than
as documents sent to the client.
# The same rules about trailing "/" apply to
ScriptAlias directives as to
# Alias.
#
ScriptAlias /cgi-bin/ "/var/www/cgi-bin/"
ScriptAlias /fcgi-bin/ "/usr/local/tipic/bin/"
Alias /txn "/usr/local/tipic/bin/tipic"
Alias /txnt "/usr/local/tipic/bin/tipic"

#
```

```
# "/var/www/cgi-bin" should be changed to
whatever your ScriptAliased
# CGI directory exists, if you have that
configured.
#
AddHandler cgi-script .cgi
<Directory "/var/www/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>

#
# FASTCGI
#
AddHandler fastcgi-script .fcg .fcgi .fpl
<Directory /usr/local/tipic/bin>
    AllowOverride None
    Options None ExecCGI FollowSymLinks
    Order allow,deny
    Allow from all
    SetHandler fastcgi-script

    <Files tipicconfig>
            AuthName "Tic Configurator"
            AuthType Basic
            AuthUserFile
"/usr/local/tipic/conf/ticconfiguser"
            Require user ticconfig
        </Files>
</Directory>


#
# Fast CGI configuration
# Fast CGI directives
#

# Timeout for responses from FastCGI
applications, including TIP
# Internet Connector

# Path to the TIP Internet Connector
configuration file.  Full path
# and file name must be supplied
FastCgiConfig FastCgiConfig -idle-timeout 40\
```

```
    -initial-env
TIPICCONFPATH=/usr/local/tipic/conf/tipic.conf\
    -initial-env
LD_LIBRARY_PATH=/usr/lib:/usr/local/tipic/lib:

#
#Additional FastCGI directives will be used
such as:
#
#FastCgiConfig -multiThreshhold 50
#FastCgiConfig -singleThreshhold 0
```

E.2. Building Apache WWW Server

Required Files

apache_2.x.tar (www.apache.org)

mod_fcgid-2.3.4.tar (www.apache.org)

fcgi-2.4.0.tar (www.fastcgi.com)

Unpack tar files

Unpack apache_2.x.tar to APACHE_SRC (eg: /usr/local/src/apache2) directory.

Unpack mod_fcgid_2.3.4.tar to APACHE_SRC/src/modules/fastcgi

Build Apache

Run: ./configure –prefix=APACHE_DIR –activate-module=src/modules/fastcgi/libfastcgi.a

Note: Replace APACHE_DIR with the directory you want apache to be installed in (eg. /usr/local/bin/apache)

Run: ./make

Run: ./make install

Configure Apache

Please refer to D.1.

Appendix F

Converting MCS Screen Formats

TIP Internet Connector version 1 maintains two locations for data definition (.t3i) files, and HTML templates (.htx files). They are specified in the TIP/ic configuration file.

The tool that maintains MCS screen formats on UNIX, tfd, does not currently handle static screen format conversion. Changes to tfd required to update this tool are outside the scope of this project. However, that does not mean that users are not able to use TIP Internet Connector efficiently.

There is a simple command line tool convt3z that shares MCS conversion library with TIP Internet Connector. This tool can be used to generate .htx/.t3i files from screen formats.

Also TIP Internet Connector itself can be used to generate ./htx/.t3i files at run-time. These files can then be used as a started project for a full conversion. Assuming that TIP Internet Connector is configured with the following values for HtxCache and T3iCache, as per tipic:

```
<Cache value="/usr/local/tipic/cache">
<RegularStore value="/usr/local/tipic/store">
<AdditionalHtml value="addhtml.htx">
```

the user can start the transaction using the screen formats he needs, and .htx and .t3i files will be dynamically created in the htx_cache and t3i_cache subdirectories, respectively.

Note: To be able to navigate between screens, the user should make sure that the addhtml.htx file contains some buttons, most likely equivalents of Transmit and MessageWait (again this is part of the TIP Internet Connector configuration).

Once that necessary files are cached they should be moved to the location where they can be updated using an HTML editor. Files in the cache location expire when not used for a long time, and updated files should be moved to a permanent location.