IoTSIP Product Description

Revision 1.2

Table of Contents

1. Introduction

IoTSIP has been created to meet the rapidly expanding needs of the IoT market to find, connect and communicate with remote devices and services.

Today, there are many ways devices can find services and find servers. The other way around is the problem. How can a software service update a device when it doesn't know the device's IP address? How does a data collection service get information from a job when it doesn't know which device is running the job, yet alone, its address?

VOIP (Voice over Internet Protocol) and applications such as Skype use the open standards internet protocol SIP (Session Internet Protocol) or customized versions to solve this problem.

This is possible because:

1) SIP uses a secure, open standards registrar where the devices themselves keep the registry up to date: there is no need for devices and services to track or hunt for the right IP address
2) SIP addresses are written in user@domain.tld format in a similar fashion to an email address making a human friendly structure suitable for a wide range of business and consumer applications

SIP Registration = DNS for dynamic addresses,
SIP Registration = DNS for the IoT

With no change to the SIP Registrar and a standardized change to the client protocol, SIP has been extended to enable SIP to find, connect and establish a DATA (rather than voice or video) connect between devices/services and other devices/services.

This product, IoTSIP, provides the client software to use the SIP internet protocol to:

- Find the IP address of a desired contact
- Establish a data connection, and
- Communicate with that device or service.

Uses of this protocol include:

- IoT devices that the OEM needs to track and with which they communicate – this structure provides secure, scalable, device self-managing registration.
- IoT devices that need to communicate with other devices, services, or micro services as this structure provides secure, scalable, device self-managing registration with human friendly nomenclature for set-up.
- Services that need to contact IoT devices to find, connect, and exchange data in a secure way using open standards and human friendly nomenclature.

This product enables IoT devices and IoT services to contact IoT devices and services with user friendly nomenclatures such as:

- Weathersensor1@farmco-op.org
- Buldozer37@Kewiet.com
- BarrFarmCombine@AGCOCORP.com

IoTSIP has applications in OEMs who need to scale their internal system for communicating with their products (IoT devices). IoTSIP also has applications as these devices are opened up to communicate directly with customer, association, 3rd party devices and services – particularly where standards and security are required.
Users include the IoTSIP product as component in their applications to aid or accomplish:

- IoT end-point discovery
- Data channel connection
- Data transfer

IoTSIP is a complete, ready-to-use, fully functioning component that includes five parts:

1. A SIP stack that follows our standard.
2. An application that uses the SIP stack to discover, connect, and transfer.
3. A channel wrapper that allows the application to control a channel.
4. A channel consisting of a Data Protocol plus Channel Protocol to do the data transfer.
5. A communications application that serves the purpose of the IoTSIP component (for example: transfer files).

Each of these parts is included in the product package. The first three are required, and, if desired, users can supply their own Data and Channel Protocols and File Transfer Protocol.

Terminology in the IoT world today is confusing. Different sources refer to different layer models. They use the same terms to mean different things and different terms to mean the same thing. To pin down what different terms mean in this document, TeleSoft International uses the OSI Model to organize the discussion of the elements of IoTSIP layers.

Words in this document with special meaning have links to and are defined in the glossary.

2. IoTSIP Components

2.1 Modified CompactSIP

The SIP stack we provide is a modified version of TeleSoft International ’s Compact SIP. CompactSIP was designed to work with media (voice/video). The modifications made were to remove voice/video and add data.

The functions available in the CompactSIP stack can be called directly to create an application extension.

2.2 CompactSIP Application

This is a TeleSoft International's proprietary application that uses the CompactSIP library to accomplish the IoTSIP functions of finding and connecting endpoints. It is compiled into SIPIoTDiscovery.so binary library.

Note: this code has **nothing** to do with passing data through the data channel. It **only** finds and connects.

2.3 Channel Wrapper

This is TeleSoft International's proprietary code allows the IoTSIP product to control a Data Channel. It is compiled into the binary library.

2.4 Channel Object

This is a Data Channel. It is provided as a binary, and a skeleton version provided as source code. You can use this skeleton to create your own channel object.

2.5 Application

This is a communications application that uses SIPIoTDiscovery.so + other objects to accomplish the goal.

3. Product Architecture

The complete IoTSIP components includes three communications functions which are describe in detail below:

- Find
- Connect
- Communicate

As a ready to use component, IoTSIP includes all the elements to Find, Connect and Communicate IoT data. The graphics below displays the elements required to Find, Connect and Communicate in the IoTSIP architecture.

3.1 Find Function

The Find function is accomplished via the SIPIoTDiscovery.so Library that includes three parts:

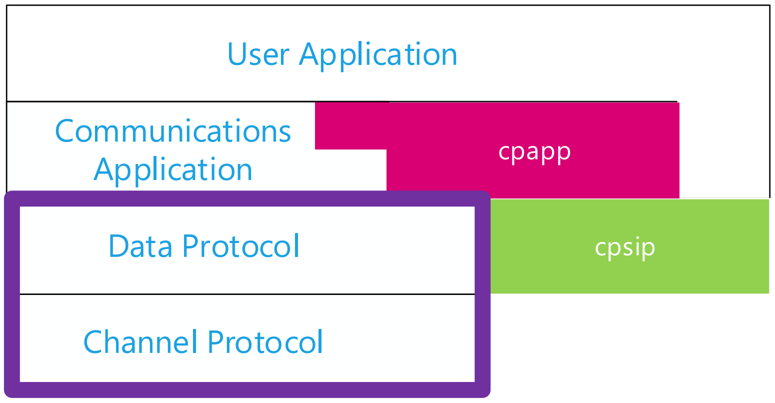
1. The SIPIoTDiscovery.so library
2. An application that uses the library to accomplish the task of the IoTSIP product
3. A wrapper that allows IoTSIP to control a data channel

This would be **all** that was necessary to provide the discover function.

This is a viable component if you already have an application with channels and a communications application, and only need to discover the end-points.

You can link your application to the SIPIoTDiscovery.so Library and add the code to make the necessary cpapp calls to discover the device URL, and use that URL to open a channel and pass data.

In the diagram below, the components of the SIPIoTDiscovery.so Library are shown in color.



- cpsip is the CompactSIP library
- cpapp is the IotNetDC application
- The purple box is the wrapper

3.2 Connect Function

If you also need to connect, IoTSIP includes a Data Channel object to control.

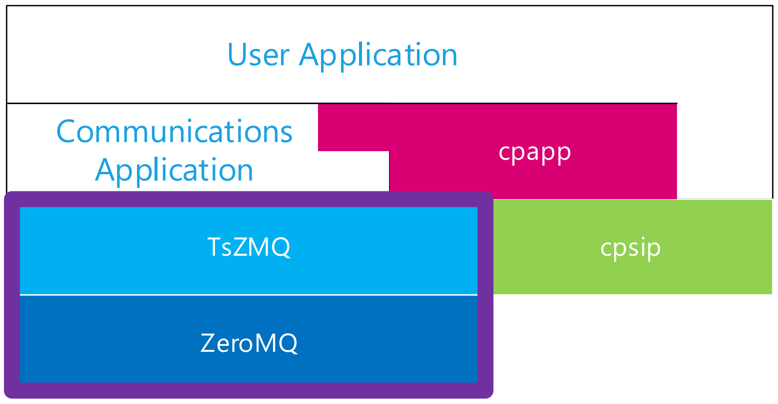
We provide documentation giving instructions to take an existing Data Protocol plus Channel Protocol and modify them so that they work as a Data Channel Object within the Channel Wrapper in the SIPIoTDiscovery.so Library. IoTSIP also provides a skeleton for the customer to use as an example/guide.

IoTSIP includes two Data Channel objects:

TsZMQ + ZeroMQ
MQTT + WebSocket

It is important to understand exactly where each of these pieces comes from. TsZMQ is a TeleSoft International's proprietary product. ZeroMQ is open source software. MQTT is a standard data protocol, and IoTSIP uses an open source implementation. WebSocket is a standard channel protocol, and IoTSIP uses an open source networking product called Mongoose to provide it.

The graphic below shows IoTSIP, including the TsZMQ channel and ZeroMQ data protocols.



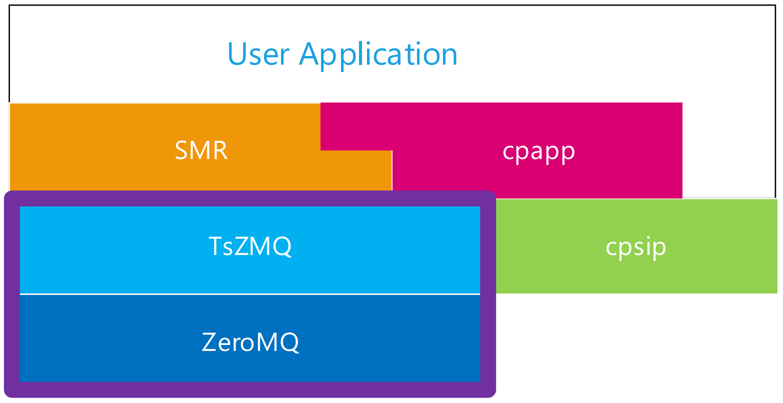
- The purple box is the wrapper
- cpsip is the CompactSIP library
- cpapp is the IotNetDC application
- The purple box is the wrapper
- TsZMQ is the TeleSoft proprietary data protocol
- ZeroMQ is the open source channel protocol

3.3 Communicate Function

If you also require a communications application_on the SIPIoTDiscovery.so Library and the chosen channel and data protocols, we provide documentation that explains how you can use your own communications application.

IoTSIP provides a communications application, the SMR File Transfer Protocol.

The graphic below shows the elements of the complete, ready to use product:



- SMR is the TeleSoft proprietary File Transfer protocol
- cpsip is the CompactSIP library
- cpapp is the IotNetDC application
- The purple box is the wrapper
- TsZMQ is the TeleSoft proprietary data protocol
- ZeroMQ is the open source channel protocol

4. Packaging

When purchasing IoTSIP, you will receive the following components:

4.1 The SIPIoTDiscovery.so binary library

This is the basic product compiled as a Linux dynamic library.

4.2 At least one channel object

TeleSoft International offers two channel objects today which can be complied as a binary, as source code, or included in the basic binary library. TeleSoft International also packages the SMR proprietary channel object as binary, and the MWT standard object as source code to give the user an example.

4.3 The worker.c file

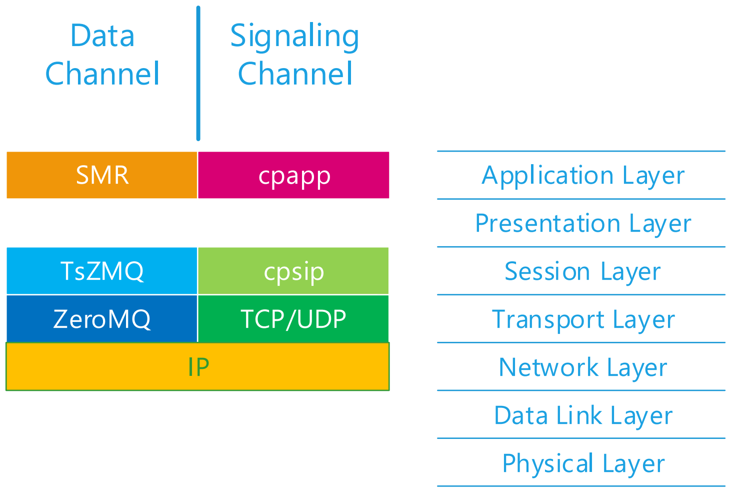
The worker.c file contains the API to IoTSIP and shows how your application can work with the IoTSIP component.

Worker.c can be used in two ways:

- You can add your code to create a complete application.
- You can add the code from worker.c into your application.

5. OSI Model

The Telecom world uses a single layer model: the OSI model. The Internet world uses an infinite number of them. As an attempt to bring clarity, IoTSIP is shown "force fitted" into the OSI model.



Note: The Presentation Layer is empty, and not needed.

5.1 SMR Data Channel Example

The SMR, proprietary file transfer communications application sends files via the proprietary TzZMQ Session layer through a ZeroMQ Transport layer over IP (the network).

The key idea is layers and frames:

SMR sits on top of TsZMQ. It breaks a file into frames, then puts a header in front of each frame to tell the TsZMQ layer on the **other end** how to handle it. It then sends it down to the **local** TsZMQ layer with instructions on how to handle it locally.

TsZMQ sits on top of ZeroMQ. It adds a header to the frame it received to tell the ZeroMQ layer on the **other end** how to handle it, then sends it down to the **local** ZeroMQ layer with instructions on how to handle it locally.

ZeroMQ sits on top of IP. It adds a header to the frame it received to tell the IP layer on the **other end** how to handle it, then sends it down to the **local** IP layer with instructions on how to handle it locally (i.e. where to send it – here is where the IP address we found gets used).

Finally, physical layer passes the data to the other end.

6. Security

IoTSIP is built with several layers of security:

1. The SIP standard architecture securely connects using the standard SIP protocol
    - It is both secure and easily scalable – works with any SIP Server (e.g., OpenSIPS)
2. IoTSIP connects through the channel protocols that have their own levels of security built in
3. IoTSIP passes data through data protocols that also have their own security built in.

Presently TeleSoft International offers two Channel Objects (combined channel and data protocols)

	Proprietary TeleSoft International Data Protocol with open source Channel Protocol	Open source Data and Channel Protocols
Data Protocol	TsZMQ is a ZeroMQ application. See below.	MQTT
Channel Protocol	ZeroMQ	WebSocket

To learn more about the security built into each of these protocols, go to:

- MQTT
    - http://www.hivemq.com/blog/introducing-the-mqtt-security-fundamentals
- WebSocket
    - https://devcenter.heroku.com/articles/websocket-security
- ZeroMQ
    - http://zeromq.org/topics:encryption
    - http://curvezmq.org/
- TsZMQ

  This is TeleSoft's proprietary use of ZeroMQ. It has same security component as ZeroMQ (which uses curve). There is a file transfer communications application (the core of SMR) that sits on top of a proprietary ZeroMQ application (TsZMQ) that uses the channel protocol ZeroMQ.

7. Glossary

*"When I use a word," Humpty Dumpty said, in rather a scornful tone, "**it means** just what I choose it to **mean** - neither more nor less."*

7.1. Communications Application

When using the OSI Model, the "Application Layer" is a communications application: an application that accomplishes some communications purpose (i.e. transferring files).

It is not to be confused with the User Application. The User Application uses the Communications Application (to transfer files).

7.2. Component

We are a part, not the whole. We are a component of a larger application.

7.3. Connection

A Connection is a Data Channel.

7.4. Data Channel

User data (i.e. a file) is sent through a data channel. We use the term Data Channel to mean a Data Protocol plus a Channel Protocol:

User data (i.e. a file) is sent through the channel using the Data (Session) Protocol format.

The Data Protocol is in turn sent through the channel using the Channel (Transport) Protocol.

7.5. Library

A collection (package) of functions. We provide it as a binary file that the user application links to.

7.6. Object

In object-oriented programing, objects are complete in themselves: they have everything they need (both code and data) to do their job.

7.7. Protocol

Wikipedia defines Protocol as:

- Communications protocol, a defined set of rules and regulations that determine how data is transmitted in telecommunications and computer networking

It is an agreement between layers on how the data is formatted, either across a connection, or from layer to layer up/down the OSI Model.

Using the Post Office metaphor:
- Users on each end agree to write in English: the protocol is English.
- The user passes the letter to the post office in an envelope with a send address, return address, and stamp: the envelope is the protocol that tells the post office where to send the letter.

7.8. Session

A Session is a conversation between two end-points. The Session Layer uses the Data Protocol to carry out the conversation.

7.9. Transport

A Transport is the method data is transferred between end-points. The name of the protocol used is also as the Channel Type.