ARMY RESEARCH LABORATORY

Terra Harvest Open Architecture Standard for Unattended Systems

by Robert Winkler, Larry Tokarcik, and Timothy Gregory

ARL-TR-6674 **September 2013**

NOTICES

Disclaimers

Army Research Laboratory

Adelphi, MD 20783-1197

ARL-TR-6674 **September 2013**

Terra Harvest Open Architecture Standard for Unattended Systems

Robert Winkler, Larry Tokarcik, and Timothy Gregory
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
September 2013	Final	

4. TITLE AND SUBTITLE
Terra Harvest Open Architecture Standard for Unattended Systems

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Robert Winkler, Larry Tokarcik, and Timothy Gregory

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: RDRL-CII-B
2800 Powder Mill Road
Adelphi, MD 20783-1197

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-6674

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Realizing adaptive and efficient use of unattended assets within a dynamic, low-bandwidth, intermittently connected network typical of a tactical environment is a challenging problem. Assets are any sensors or platforms that can be tasked and/or reconfigured to produce a data payload, which is then disseminated over a communications channel. Discovery of assets, moving payloads from assets through gateways, and archiving, processing, and harvesting payloads for remote analysis are key capabilities of a foundation for an adaptive architecture. Such an architecture enables setting in motion payloads that can be used both locally and globally when they come to rest within a domain-specific data store. This report describes the Terra Harvest high-level architecture components for acquiring, persisting, processing, and disseminating information from unattended assets to produce mission-specific behaviors and information.

15. SUBJECT TERMS
Unattended ground sensors, UGS, asset discovery, OSGi, Terra Harvest

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
					Timothy Gregory
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (*Include area code*)
Unclassified	Unclassified	Unclassified	UU	20	(301) 394-5604

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

1. Introduction

The Terra Harvest Open Architecture Standard for Unattended Systems is a Defense Intelligence Agency (DIA) sponsored specification for a software framework facilitating asset interaction in a plug-and-play manner. A fundamental objective of the Terra Harvest project is to provide a software architecture specification that will provide this flexibility while still effectively operating within a size, weight, and power (SWaP) constrained environment. An initial Terra Harvest reference implementation called Terra Harvest Open Source Environment (THOSE) based on these functional requirements built on the Java programming language and the Open Service Gateway Initiative (OSGi) framework has been developed and tested at Trident Spectre 2012 and 2013. OSGi was chosen for the reference implementation because of its ability to run on Linux or Windows, its acceptance in the industrial and commercial communities (mobile phones, automobiles, industrial automation, building automation, entertainment, and fleet management), and its small footprint, which specifically addresses the Terra Harvest SWaP objective.

Even through the Terra Harvest reference implementation is focused on an embedded environment, it will scale from the embedded world up to the laptop, workstation, and server level platforms where SWaP is not an issue. It should be noted that Terra Harvest is specifically designed for rapid integration of different assets and is not intended as a framework for a system of systems where other commonly accepted industry practices and approaches are more appropriate.

2. Terra Harvest Controller Configurations

Terra Harvest provides the functional requirements foundation for configuring an unattended system that when connected to assets (sensors, communication devices, processing elements, etc.) can provide a specific mission capability. A Terra Harvest Controller (THC) is the hardware platform that hosts a Terra Harvest software implementation, along with asset-specific plug-ins and services. The use of the Java programming language and OSGi for the implementation of Terra Harvest allows a controller to be realized on many different computing platforms, from a militarized embedded processor packaged within an enclosure that will survive harsh environments to commercial laptops, workstations, or servers that serve as gateways to higher-echelon processing, dissemination, and exploitation (PED) applications. From an abstract perspective a THC is a host processing platform running a Terra Harvest implementation that when configured with the appropriate hardware and software components provides a capability for a specific mission. In its simplest form a THC connects one or more sensor and communication assets (figure 1). More concrete configurations are described below.

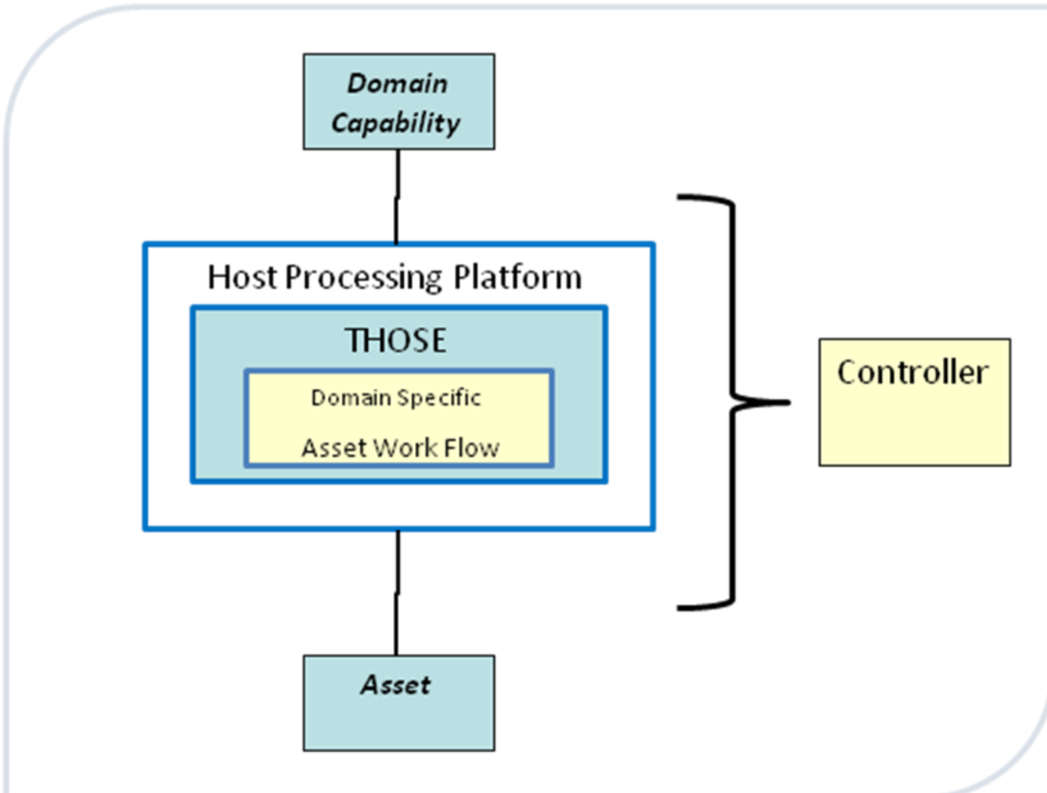

Figure 1. Abstract THC configuration.

2.1 Local Wired Asset with Local Wireless Exfiltration Configuration

In this configuration, the controller performs limited mission-specific processing onboard and functions primarily as a communications relay, which acquires data from a local sensor and then disseminates the data over a local wireless link to another local node that is designated as the central gateway controller. The gateway controller is then responsible for performing further domain-specific refinement and processing before exfiltrating the data to a domain-specific gateway node via a reach-back link (figure 2).

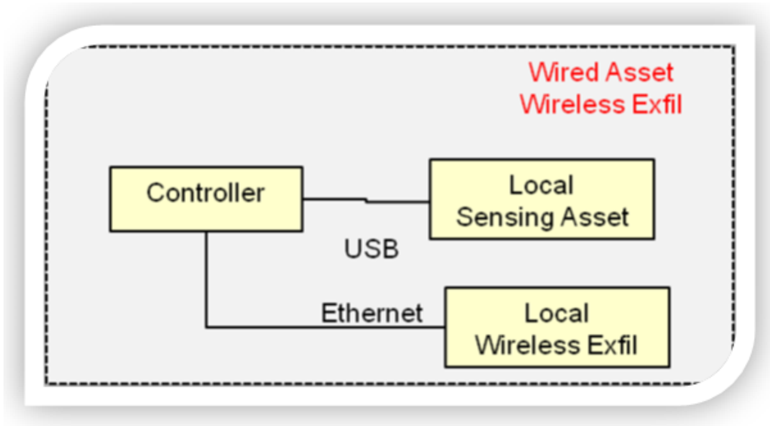

Figure 2. Local wired asset with local wireless exfiltration.

2.2 Remote Wireless Asset with Local Wired Asset and Wireless Exfiltration Configuration

In this configuration, one or more remote sensors, possibly of different modalities, are connected to the controller via a wireless link. Adding more than one sensor modality may also require additional onboard payload refinement/processing (i.e., adding time, tag, and location) before the payload can be exfiltrated (figure 3).

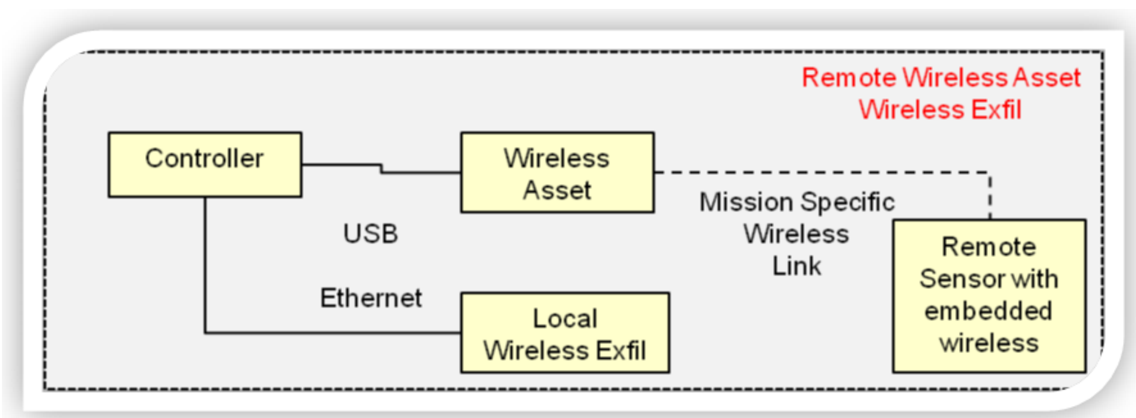

Figure 3. Remote wireless asset with local wireless exfiltration.

2.3 Remote Wireless Asset with Local Wired Asset and Wireless Exfiltration Configuration

As controller and asset deployments get more complex, the onboard payload processing will continue to grow. Onboard mission-specific processing will be added to a controller, which further defines the controller's mission behavior. A simple mission behavior may be required that only tags and sends payloads from a specific asset if a specific detection occurs with a certain time frame and ignores all others. Another mission may require onboard specific processing for profiles where one or more sensors are positioned some distance from the controller and the detections from the remote sensors (i.e., seismic sensors) will, in turn, drive the actions of a locally wired sensor (e.g., an imager). The onboard controller mission profile processing must be able to correlate and synchronize the local events in order to produce a payload at a higher operational context level (i.e., object is moving north to south), which can also be exfiltrated along with the raw sensings (figure 4).

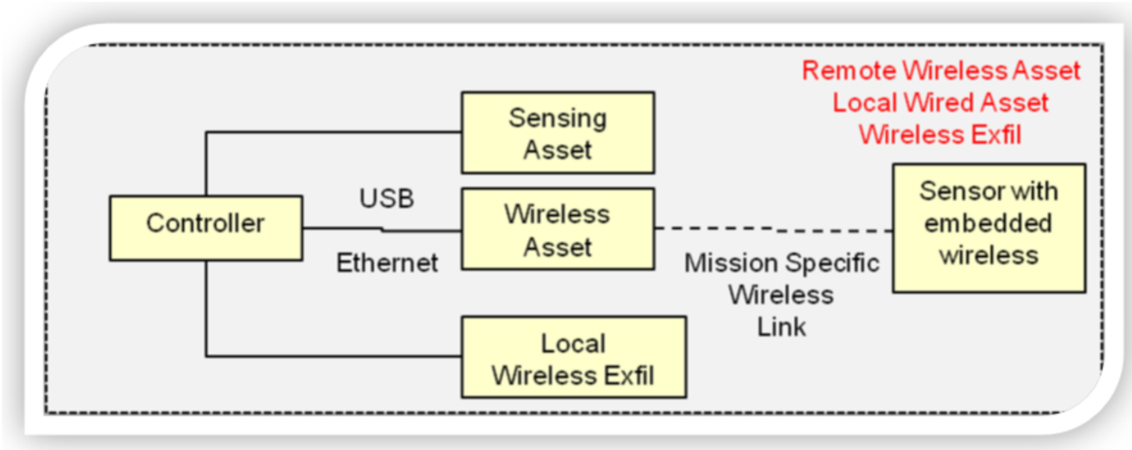

Figure 4. Remote wireless asset with local wired asset and wireless exfiltration.

2.4 Local Gateway Configuration

In this configuration, multiple controller nodes operating within a local area exfiltrate their payloads to a controller designated as the local gateway controller. The local gateway controller is configured like any other local controller. It can have sensors connected to it through wired and/or wireless communication links in the similar manner as any other local controller. The local gateway controller is configured with a local wireless link that is used to ingest data from all the other local controllers. Its distinguishing characteristic is the addition of a long-haul wireless communication link used for exfiltrating the collective data of the sensor field to higher echelons. If the long-haul reach-back communications link is bidirectional, the local gateway controller also serves as the path for remote mission programming and asset fine-tuning (figure 5).

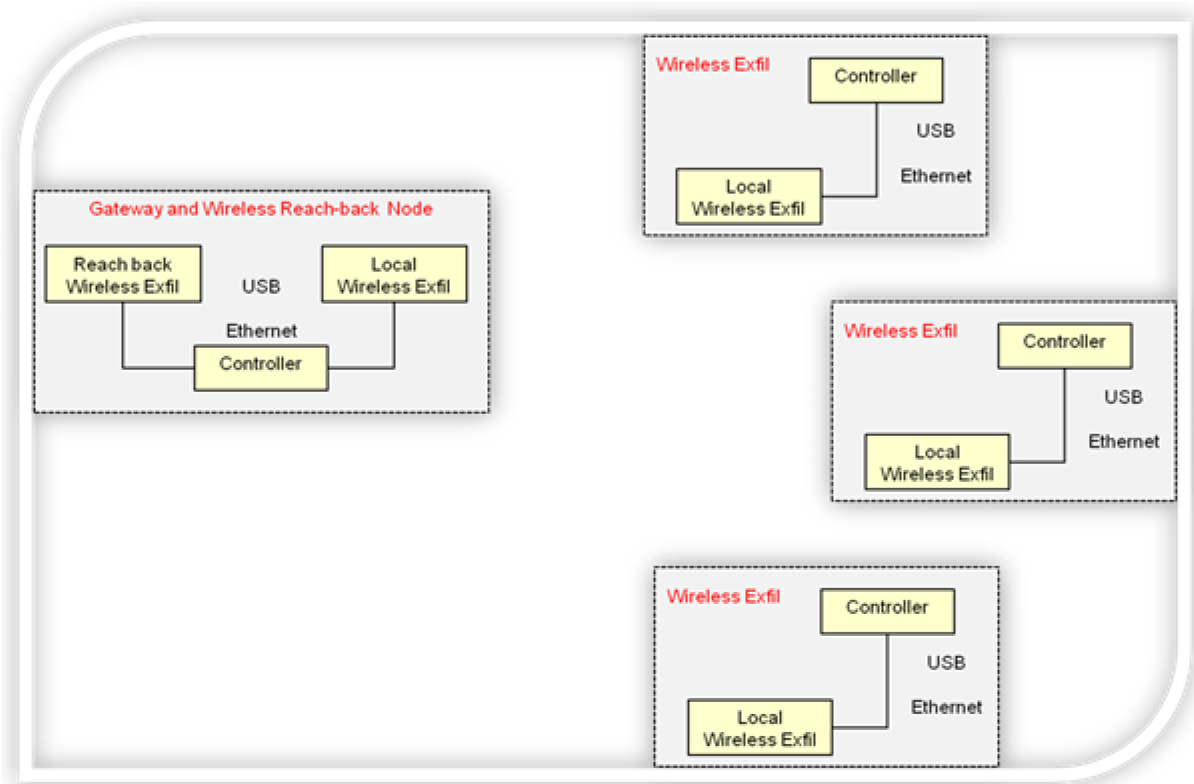

Figure 5. Local gateway.

2.5 Harvester Configuration

Gateway controllers that are positioned where reach back to higher echelons is not possible due to communication restrictions require the gateway controller to be configured as a store-and-forward system, which holds the data until another controller configured specifically as a communications relay comes into range to harvest the data.

2.6 Domain Gateway Configuration

In this configuration, the controller is typically located at a higher echelon (e.g., Tactical Operations Center or Forward Operating Base) with more infrastructure (shore power, high bandwidth wired communications links, etc.). These controllers are typically deployed on laptops or desktop workstations and serve as the bridge between the remote sensor fields and higher echelon PED applications (figure 6).

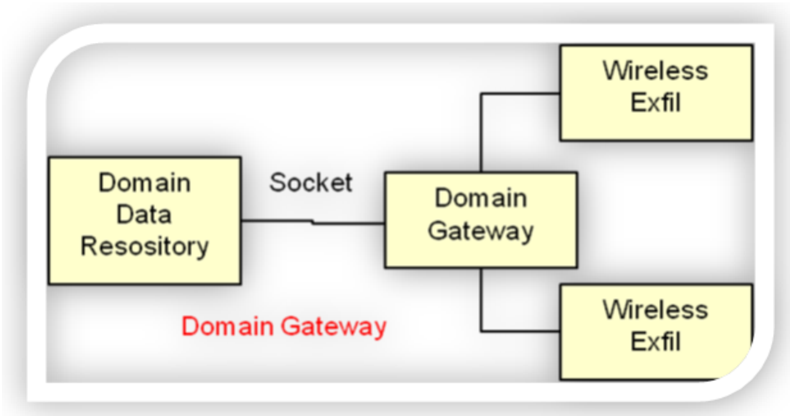

Figure 6. Domain gateway.

3. Terra Harvest Software Components and Capabilities

The major software components within the Terra Harvest software framework are shown in figure 7. Components are configured and deployed based on the current mission requirements. In the Terra Harvest, reference implementation software components are implemented as plug-ins and services that are discovered, loaded, and activated under the OSGi framework. As described above in the controller configurations section, the baseline function of a controller is to acquire data from an asset and then disseminate the payload to other controllers or another domain-specific software component. The acquisition and dissemination of data are accomplished using industry-standard (wired or wireless devices) or vendor-provided, asset-specific devices (MicroHard, Security Equipment Integration Working Group [SEIWG], Common Sensor Radio (*1*), Iridium, Broadband Global Area Network [BGAN], etc.) registered as communication services. An asset (sensor, communication, or platform device) interfaces with the Terra Harvest software infrastructure through an asset plug-in. The main role of the asset plug-in is to acquire data or send commands to an asset using the communication service assigned by the mission program. After acquiring the data from the asset, the asset's plug-in is responsible for transforming the raw data into a payload format that conforms to a industry/Government-defined standard lexicon specified as an extensible markup language (XML) schema. The asset plug-in then stores this translated payload into the Persistent Store. The Persistent Store acts as a repository for both data that are local to the asset as well as data that are made available for other plug-ins to refine, fuse, process, or disseminate. Refine/Process plug-ins pull asset-specific or multi-asset data from the Persistent Store and then either refine the payload (adding control tag meta data, time, location, tagging the payload for exfiltration, etc.) or process the data in order to add context to the raw payload (detecting an object in an image, correlating multiple lines of bearing, classifying a target, etc.). The Dissemination component is responsible for exfiltrating designated payloads off the controller using the communication service assigned by the current mission profile. The Mission Profile component is responsible for coordinating and orchestrating the assets to reflect the current mission. In the Terra Harvest reference implementation, Mission Profiles are defined in JavaScript with access to the interfaces to the other plug-ins and services within the controller.

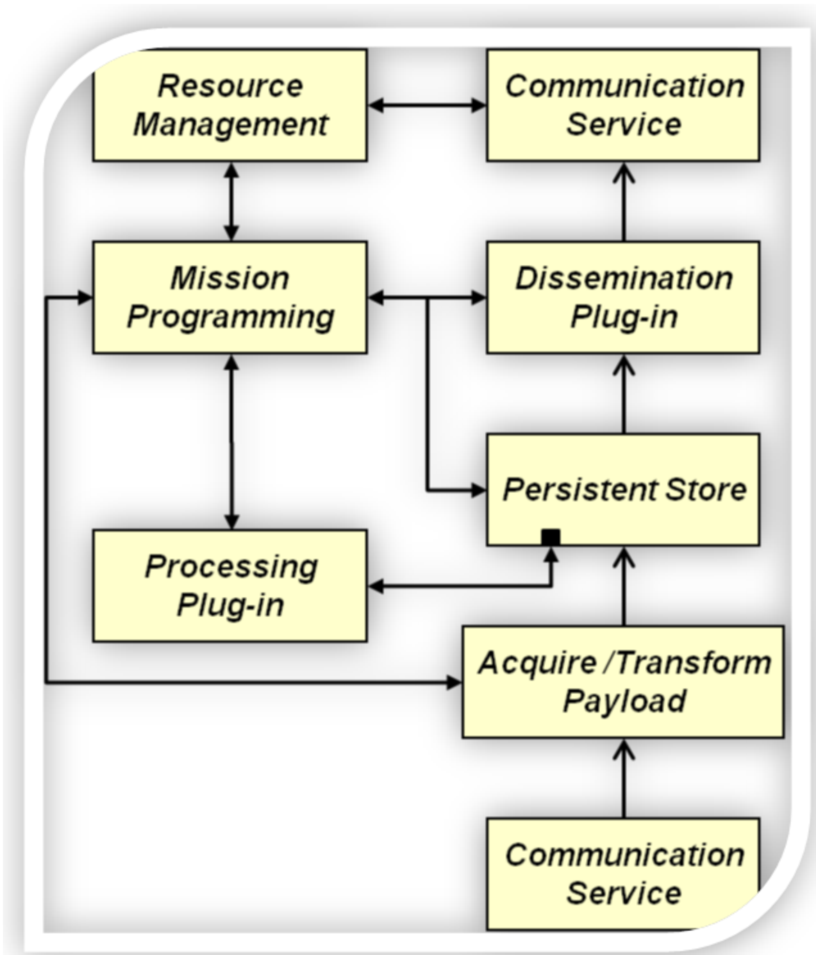

Figure 7. Terra Harvest components.

3.1 Communications Services

Communication services implement the physical, link, network, and transport layer of the Open System Interconnect (OSI) model. All assets connected to a controller at the physical level use industry-defined standard electrical (serial, universal serial bus [USB], Ethernet) interfaces and the data link software layer uses the drivers built into the operating system and the OSGi framework. The link and network layer are specific to the communication asset and is provided by the individual communication plug-ins. Payload framing and headers are two critical components required in order to move payloads from a producer to a consumer. Assets that produce payloads (i.e., sensors) must be able to package and uniquely identify their payloads before sending the payload over a wired or wireless communication service. If an asset payload size is larger than the physical link Maximum Transmission Unit (MTU) size, then the payload must be segmented using some agreed upon packet size for that physical link. If more than one MTU is required to pass the payload, then a segmentation protocol must be defined in order to reconstruct the payload on the receiving end. By establishing a standard for payload segmentation (header plus segment) that takes into account the framing of the physical link, payloads can be segmented and identified on the sender side, transmitted over the physical medium, and then reassembled on the receiving end thus ensuring payload delivery. Sensor to receiver payload delivery can be provided through the use of a common segmentation library on both the sending and receiving side of the physical link. The purpose of the segmentation library is to break up a payload into segments, which along with the additional header information, is smaller than the MTU of the physical link (figure 8).

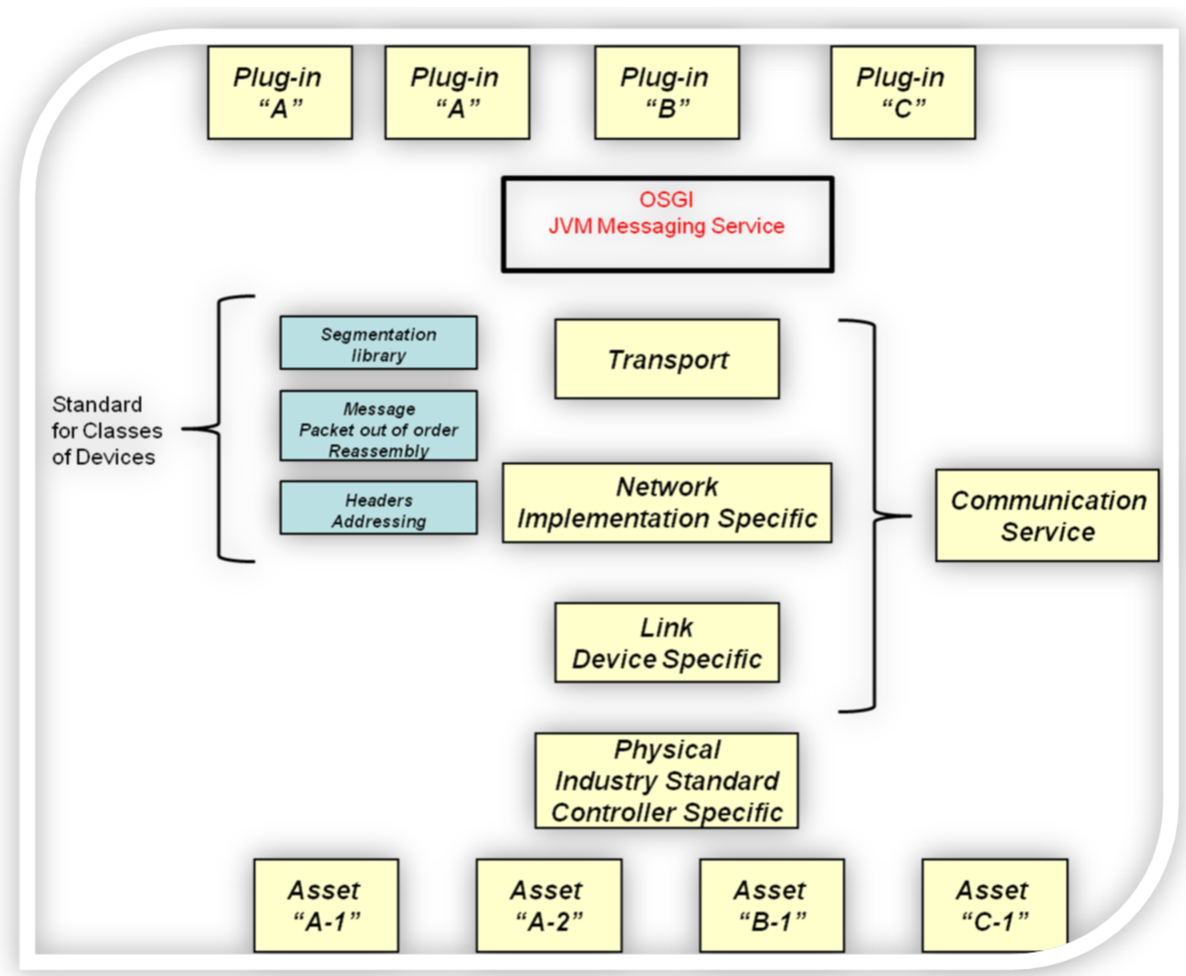

Figure 8. Communication service plug-ins.

3.2 Asset Plug-ins

Assets are data producers that are attached to a controller using a wired or wireless physical link. An asset plug-in is provided by the asset vendor and is loaded onto the controller. The primary functions of an asset plug-in are to acquire asset payloads using a communication service, perform any asset specific processing on that payload, transform the payload from asset specific format to an internal representation based upon a standard lexicon, and persist it in the Persistent Store. In the THOSE reference implementation of Terra Harvest, plug-ins use OSGi interfaces to discover and connect to the desired communication service either provided by the operating system for wired connections (i.e., USB, Ethernet) or by a communication service plug-in that provides a wireless connection off platform. The asset plug-in also uses OSGi to establish a connection to the Persistent Store component. A plug-in may also use OSGi services to connect to the Resource Management component in order to receive commands on start-up and during operation. An asset plug-in should also be able to respond to external commands that are sent to the plug-in through the OSGi event admin interface. These commands also conform to a standard command lexicon. The asset plug-in must process these commands and then either respond to them locally or further process these commands by transforming them into asset-specific workflow and payloads formats that are sent to the asset via the appropriate communication plug-in/service.

3.3 Persistent Store

The Terra Harvest Persistent Store component is the central repository for onboard storage and information brokering. The Persistent Store consists of two storage areas. The first area is a general scratch storage, which can be used by onboard plug-ins for any temporary plug-in specific storage. The other area is dedicated to temporary storage for Terra Harvest-compliant payloads, which are defined as observations (i.e., events and sensed payload data) and command data. Terra Harvest-compliant payloads are based upon an industry/Government developed and evolving common lexicon. All plug-ins must conform to the common lexicon semantics that are represented within a defined XML schema. Plug-ins can register through the Persistent Store application programming interfaces (APIs) for events and/or sensed payloads from other assets. Plug-ins can register for commands from other internal or external components through the Persistent Store interface. Using the Persistent Store as the path for event processing within the controllers enables multiple components to subscribe and respond to a single event. Event subscription and dissemination (one to one, one to many, or broadcast) is handled by the OSGi Event Admin service, removing this burden from the Persistent Store. Built on common APIs and lexicon, the Persistent Store component is a prime candidate for competitive implementation within the vendor community.

3.4 Dissemination

The Terra Harvest dissemination component is responsible for moving payloads off a controller platform to another controller or domain-specific node. The dissemination component is implemented as a plug-in. Whereas, the sensor asset plug-in registers with a communication service to acquire payloads from a wired/wireless connected sensor, processes the payload, transform the data in accordance with a common lexicon, and then persists the data in the

Persistent Store. The dissemination plug-in registers for events and payloads using the Persistent Store APIs and then either sends the payload using a communication service, which will move the payload off platform. Dissemination plug-ins can be very simple or they can contain behaviors that are designed for specific operations. The simplest form of dissemination plug-in is a simple pass-through. A simple pass-through dissemination plug-in registers for observations from the Persistent Store and then simply passes these payloads directly to the communication service for exfiltration off the platform. A simple pass-through dissemination plug-in does not perform any additional processing on the payload nor does it reformat the payload so that it can be efficiently sent over a bandwidth limited link. Dissemination plug-ins can be designed to have domain-operational-specific behaviors, such as only send this type of payload out this communication service and another type of data out this communication service. More complex dissemination behaviors can be accomplished through the mission programming component.

3.5 Processing

Terra Harvest processing components are responsible for executing any onboard payload processing. Processing components subscribe to the Persistent Store for observations of a certain type or from a particular asset (e.g., if an image processing algorithm is required to perform additional processing on images to determine if they are of adequate quality to be disseminated). Processing plug-ins can be used to correlate multiple payloads from multiple assets (e.g., correlating multiple lines of bearing in order to determine object location). A processing plug-in may produce events that are consumed by other processing plug-ins (e.g., a line of bearing correlation event may trigger a query for imagery at a location, which, in turn, is processed generating yet another linked observation). Processing plug-ins can produce events that, in turn, change the controller’s behavior and/or the behavior of assets attached to the controller.

3.6 Mission Programming

The mission programming component is responsible for providing mission-specific control of the controller and all attached assets. The mission program can consist of simply a series of standard commands and/or domain-specific functions calls implemented within a scripting language. Mission programs would be developed by knowledge end-users or vendors, and then made available in an equivalent to an application store for end-users to download, configure, and in some cases, modify for a specific mission. Mission programs will also have access to onboard specific controller functions that enable the mission program to control power, assets, event and observation processing, and dissemination. Mission programs can be as simple as an orderly initialization of the controller and its assets or a complex as dynamically changing the controller’s behavior depending on the state of events within the local and/or distributed environment. Mission programs can operate in a hierarchical fashion in which a higher level controller is maintaining state and dynamically changing behaviors of more than one controller and it assets.

4. Conclusion

This report outlined the basic services and capabilities required for an interoperable unattended systems controller, which would allow assets of a variety of different types and modalities and from a variety of different vendors to communicate with one another in a plug-and-play manner. This work could serve as a basis for further standardization for unattended systems.

5. References

1. Fisher, F.; Dice, E. *Functional Configuration Testing for the Common Sensor Radio*; ARL-TR-5832; U.S. Army Research Laboratory: Adelphi, MD, 2011.

List of Symbols, Abbreviations, and Acronyms

APIs	application programming interfaces
BGAN	Broadband Global Area Network
DIA	Defense Intelligence Agency
MTU	Maximum Transmission Unit
OSGi	Open Service Gateway Initiative
OSI	Open System Interconnect
PED	Processing, Dissemination, and Exploitation
SEIWG	Security Equipment Integration Working Group
SWaP	size, weight, and power
THC	Terra Harvest Controller
THOSE	Terra Harvest Open Source Environment
USB	universal serial bus
XML	extensible markup language

NO. OF COPIES	ORGANIZATION
1 (PDF)	DEFENSE TECH INFO CTR ATTN DTIC OCA (PDF)
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
5 (PDFS)	US ARMY RSRCH LAB ATTN RDRL CII B T GREGORY R WINKLER L TOKARCIK ATTN IMAL HRA MAIL & RECORDS MGMT ATTN RDRL CIO LL TECHL LIB