Advances in Computational Sciences and Technology
ISSN 0973-6107 Volume 10, Number 9 (2017) pp. 2765-2772

http://www.ripublication.com

An Observer Design Pattern to Implement Reuse of Secure Channel in Client Driver Using OPCUA

Meghana.K.K[1] and Lincy Meera Mathews[2]

[1]*Department of ISE, Ramiah Institute of Technology, India.*

[2]*Department of ISE, Ramiah Institute of Technology, India.*

Abstract

In this paper, observer design patters is suggested for the reuse of secure channel implementation of OPCUA client driver in java language. As we know, observer pattern is a behavior pattern it helps managing push and pull requests effectively. Supports better management and synchronization. By reusing secure channel we save lot of memory and time to connect and create a new secure channel.

Keywords: OPCUA, secure channel, reuse, java, observer pattern

I. INTRODUCTION

Open Protocol Communication Unified Architecture is being applied to many areas in manufacturing software in application areas such as Field Devices, Control Systems, Manufacturing Execution Systems and Enterprise Resource Planning Systems. These systems are designed to exchange information and to use command and control for industrial processes. OPC UA provides a common infrastructure model to assist this information exchange OPC UA specifies the information model to represent structure, behavior and semantics along with the message model to interact between applications which allow the communication model to transfer the data between end-points. OPC UA also assists the model to guarantee interoperability between systems.

OPC UA is a platform-independent standard by which various types of system(s) and

device(s) communicate by sending Messages among Clients and Servers through various kinds of networks. It supports robust, secure communication which assures the identity of Clients and Servers and avoids attacks. OPC UA provides set of Services which Servers may provide, and also individual Servers state to Clients what Service sets they support. Information is transmitted using OPC UA - defined data types, and Servers defined object models that Clients can animatedly discover. Servers will provide access to current as well as historical data, Alarms and Events will notify Clients of significant changes. OPC UA can be mapped onto a range of communication protocols where data can be encoded in many ways to trade portability and efficiency.

The observer pattern is a software design pattern in which an object, called the subject, maintains a list of its dependents, called observers, and notifies them instantly of any state changes, normally by calling one of the methods[8]. This method of reusing the same secure channel is been implemented in C++ along with HTTPS. In java you have built in interfaces and packages serving the purpose of synchronization and patters can have a better understanding and simple to design in java along with OPC/TCP or with OPC/HTPP(s). By having executor design pattern along with observer and multithreading (thread pool) we can have a track of the events. Listener implementation is easy in Java. It is easy to include more and more monitoring items in Java according to the specification OPC foundation as read in the specification document.

II. LITERATURE SURVEY

[1] This paper highlights a roadmap, having three alternative ways, to adopt OPC UA, a new generation OPC specification. The OPC UA specification looks at improving and extends the scope of OPC in system integration. It introduces SOA paradigm and platform independence into OPC world, enabling a wide variety of new applications. [2] This paper gives detail on OPC UA XML Web services mapping .This Web service provides interface to access process data. These services use XML technology for data exchange. It is very much effective in information exchange but only one drawback is that XML alone cannot provide cryptography. Hence we need to adopt java for better implementation.

[4] This paper presents the OPC UA JavaScript framework (JSUA) where we have incorporated a JavaScript implementation of the OPC UA communication stack. Time measurements showcase the performance of JSUA on different kind of devices and compared to a native OPC UA where the client is implemented in C++.[5] Part 3:

OPC UA Specification: Address Space Model

This specification provides the guidelines to understand about the server connection and address space of the server which will interact with the client and machines.[7] Part 4: OPC UA Specification: Services. This describes the service set of the OPC UA.

Service set like connection service set, subscription service set that has the guidelines to develop these services.

III. SYSTEM DESIGN

A Secure Channel is a long-running logical connection between a single Client and a single Server. This channel maintains a set of keys known only to the Client and Server, which are used to authenticate and encrypt Messages sent across the network. The Secure Channel Services allow the Client and Server to securely negotiate the keys to use.

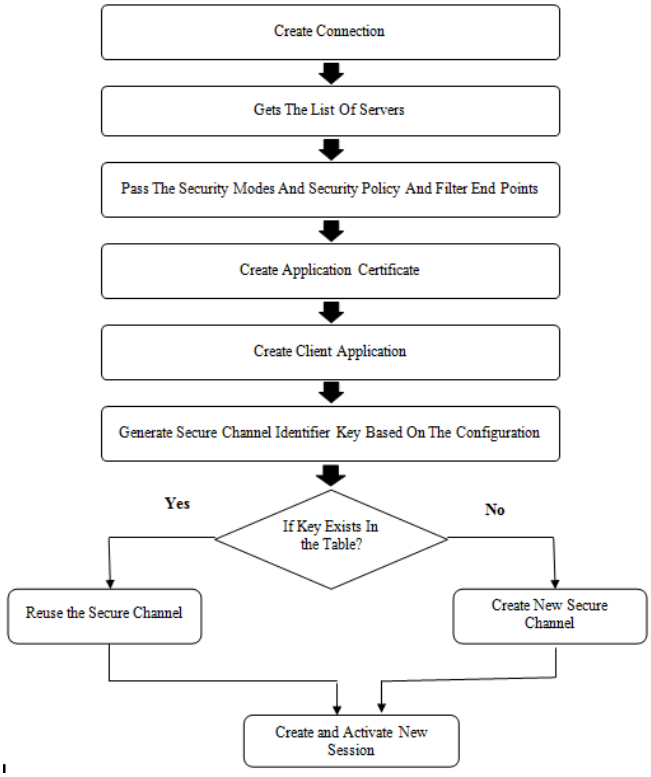

Fig. 1: Data flow diagram of reusing secure channel

Creating connection is main thread which initiates the create secure channel then followed by create session . For single client, one or many connection request to single OPC UA server with same connection properties/configuration, will result in only one securedChannel creation. This single securedChannel object is used across all relevant connections. In case, any of client, server or connection properties changes, and server connection request will result in creation of another securedChannelobject.

Secured Channel creation being expensive operation, optimization has been achieved

with the approach. Driver is designed to let only single request of concurrent requests to create secured channel object and keeping other requests blocked till secured object is created and activated. Post creation same secured channel object reference is returned to all the blocking requests.

However in case of session creation, always a new session object per get Server Connection request is created.

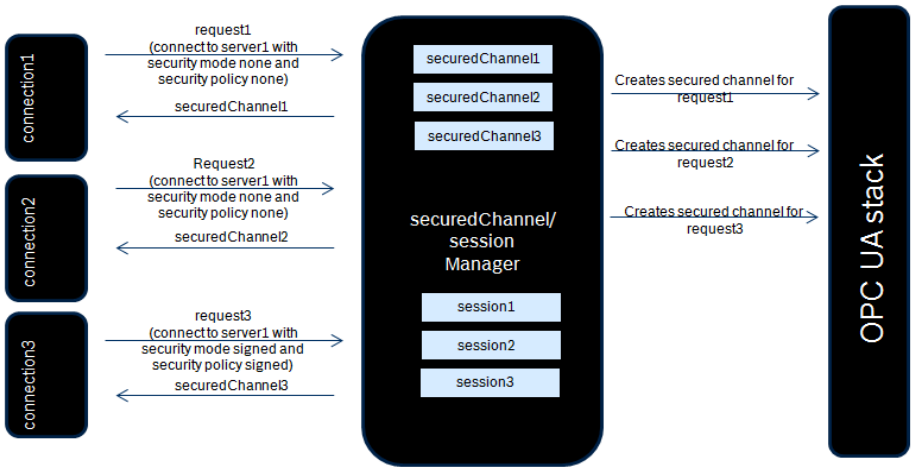

Fig. 2: Activity diagram of creating secure channel without reusing concept.

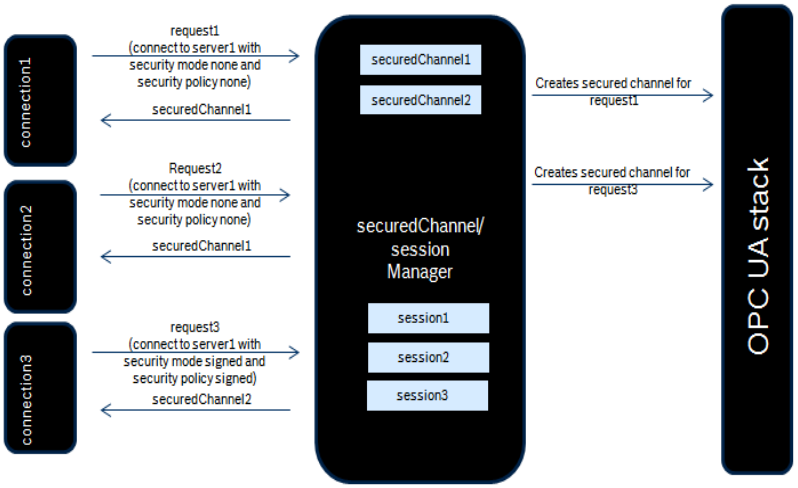

Fig. 3: Activity diagram of reusing secure channel.

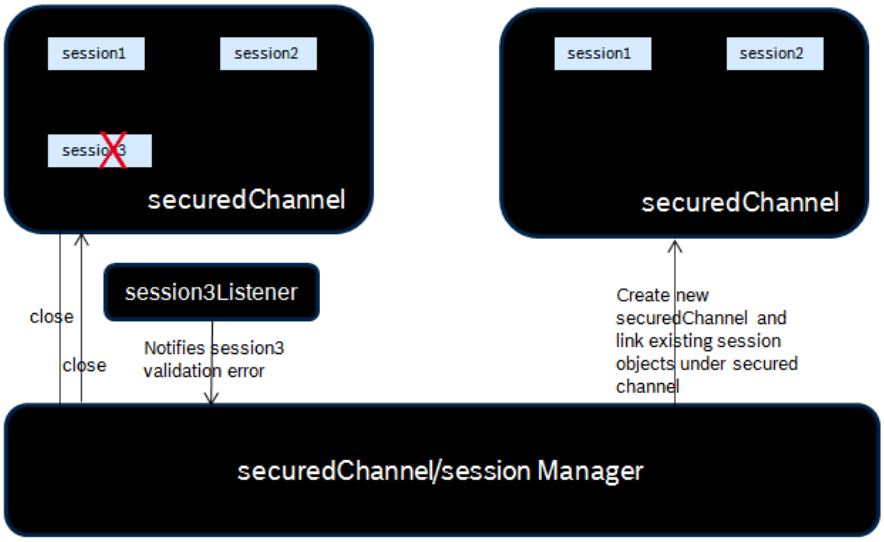

Fig 4: Activity diagram of listener implementation.

Security concerns are taken through various validations like certificate validation, validating response body etc. under connection object transparent to end user.In case of validation error like attempt to modify response, the current session is destroyed along with securedChannel carrying the session. A new secured channel is created with same configuration as before. All existing session objects under secured channel destroyed, gets transferred under new secured channel created.

Here we see that in Fig.3 there is a creation of secure channel for each of the same connection requests(Connection-1,Connection-1,Connection-2). There is a creation of 3 secure channels. Where we could have used the same secure channel 1 for connection-1 request- 2. This is made possible in Fig.4 where there is only 2 secure channel and 3 requests. Here the channel is being reused. We have only 2 secure channel for 2 connections Fig.4 shows the listener used to notify the events under these reuse of secure channel.

V. RESULTS

Scenario is that we are having connection-1 which is requesting for one session. We name it as session-1. Connection management is going to create a secure channel-1 for the request. Again connection-1 requesting for another session named it as session -2. The listener will intimate the connection manager that already this connection exists and the secure channel can be reused. Connection manager checks the table for Key and authentication ID(session ID). If it hits the key value pair it allows the connection -1 request -2 to access and communicate through secure channel -1.

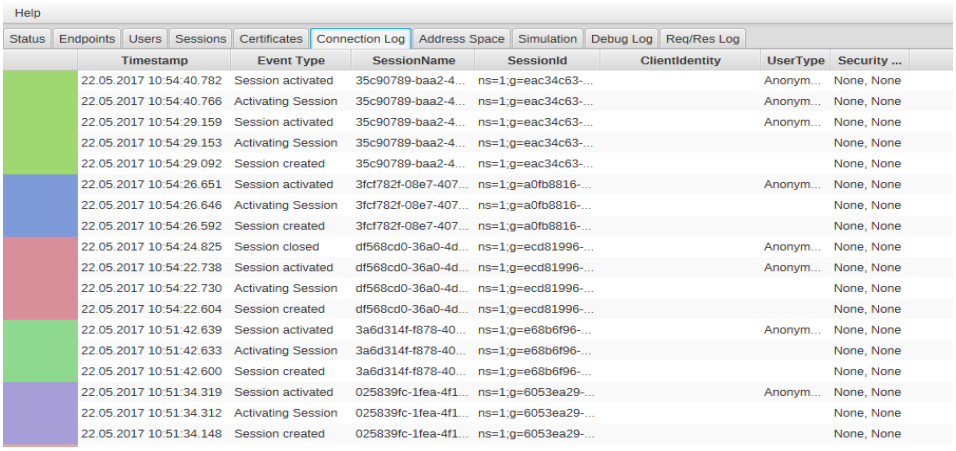

Fig. 5: Connection log

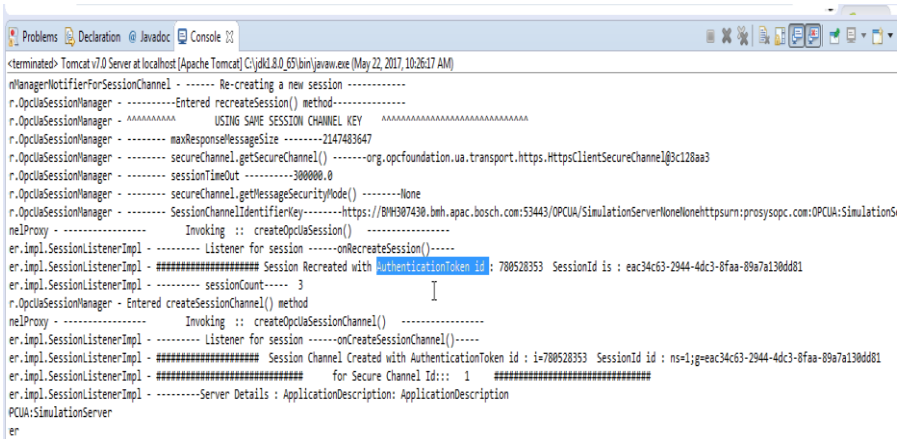

Fig. 6 Screenshot of same secure channel ID (session 1- connection 1)

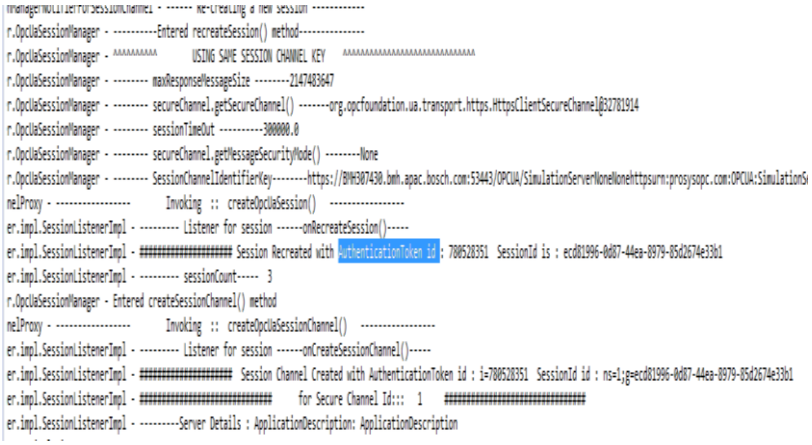

Fig. 7 Screen shot of same secure channel ID (session 2 connection 1)

Here in results Fig 5 we have connection log of connection-1 having the session ID and under same secure channel many sessions are being created. Green color shows that same secure channel has been used for all the sessions under that connection. The same is verified in the program output in eclipse. The listener is listening to the connections made and referring to the table to report to the observer if any change in the connection it is being noticed and asked the connection manager to reuse the same secure channel.

V1. CONCLUSION

Listener(Observer pattern) helps in observing the connection activities and keep track of the session ID and secure channel that has been created. It also alerts the user when the session has been tampered. In case of session close abruptly then it is intimated to the observer(Subject). Another secure channel with same credentials is being created and sessions are being transferred. Hence this patter is suggested to use in the implementation of reuse of secure channel.

REFERENCES

[1] T. Hannelius, M. Salmenpera and S. Kuikka, "Roadmap to adopting OPC UA," *2008 6th IEEEInternationalConference on Industrial Informatics*, Daejeon, 2008, pp.756-761.doi:10.1109/INDIN.2008.4618203.

[2] A. Braune, S. Hennig and S. Hegler, "Evaluation of OPC UA secure communication in web browser applications," *2008 6th IEEE International Conference on Industrial Informatics*, Daejeon, 2008,pp.1660-1665.doi:10.1109/INDIN.2008.4618370

[3] R. Huang, F. Liu and Pan Dongbo, "Research on OPC UA security," *2010 5th IEEEConference onIndustrial Electronics and Applications*, Taichung, 2010, pp. 1439-1444.doi:10.1109/ICIEA.2010.5514836

[4] M. Freund, C. Martin, A. Braune and U. Steinkrauss, "JSUA — An OPC UA JavaScript framework," *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation(ETFA)*,Cagliari,2013,pp.1-4.doi:10.1109/ETFA.2013.6648128

[5] Part 3: OPC UA Specification: Part 3 – Address Space Model http://www.opcfoundation.org/UA/Part3/

[6] E. Moraes *et al*., "Improving connectivity for runtime simulation of automation systems via OPC UA," *2015 IEEE 13th International Conference on Industrial Informatics(INDIN)*,Cambridge,2015,pp.288-293.doi:10.1109/INDIN.2015.7281749

[7] Part 4: OPC UA Specification: Part 4 – Services

[8] http://www.opcfoundation.org/UA/Part4/

[9] A. Maka, R. Cupek and J. Rosner, "OPC UA Object Oriented Model for Public Transportation System," *2011 UKSim 5th European Symposium on Computer Modeling and Simulation*, Madrid, 2011, pp. 311-316. doi:10.1109/EMS.2011.84

[10] Wikipedia.org https://en.wikipedia.org/wiki/Observer_pattern

[11] https://sourcemaking.com/design-patterns-ebook

[12] White Paper – OPC UA Publisher/Subscriber Communication with TSN Ethernet.