# SmartSlog Session Scheme for Smart-M3 Applications

Aleksandr Lomov
Petrozavodsk State University (PetrSU)
Petrozavodsk, Russia
lomov@cs.karelia.ru

**Abstract**

Smart spaces provide a shared view of dynamic resources and context-aware services within a distributed application. Smart-M3 is an open-source platform that implements the smart space concept. It inherits the tuplespace and publish/subscribe asynchronous communication models, and agents (Smart-M3 knowledge processors, KPs) cooperate sharing the common smart space. A Smart-M3 semantic information broker (SIB) maintains the smart space in low-level ontological terms of triples. Smart Space Access Protocol (SSAP) is used for session management and triple governance transactions. This paper considers the session between KP and SIB in the Smart-M3 applications. First, SSAP session will be studied. Second, a new SmartSlog session scheme will be introduced. It operates with such ontological concepts as a class, a property, and an individual and extends the capabilities of SSAP session. New session scheme is based on SSAP session and can manipulates with many SSAP sessions by combining them in one session. All data are shared between SSAP sessions and it allows to work with different smart spaces as with one on KP side. The new session scheme is partially implemented in SmartSlog ontology library.

**Index Terms:** Smart spaces, Smart-M3, Session, SSAP, SmartSlog.

## I. Introduction

Smart spaces provide an environment for heterogeneous devices and programmable agents to share their resources and services [1], [2]. Smart-M3 is an open-source interoperability platform for information sharing [3]. Smart-M3 provides a multi-agent distributed application with a common smart space to share dynamic knowledge and to make reasoning cooperatively. It allows to implement information subsystems of smart environments [4], [5]. A Smart-M3 semantic information broker (SIB) maintains its smart space and represents the content in such low-level terms as RDF triples. A Smart-M3 application consists of knowledge processors (KPs) that share the application smart space. Since the same SIB can be used by several application, the SIB supports several application smart spaces within its own space.

A Smart-M3 application consists of knowledge processors (KPs)—distributed agents running on various computers and participating in the smart spaces. A smart space stores information in a shared RDF triple store. Each KP connects to a SIB using the Smart Space Access Protocol (SSAP). The developer of the KP logic uses a knowledge processor interface (KPI) to smart spaces. Up to perform operations on the SIB the KP must start a SSAP session. KP can work with several SIBs simultaneously, but each connection will be held in separate session. There is no mechanism to work with many SIB in one Smart Space session. To achieve this, the developer should implement a session management and data storage which can share data between all SSAP sessions.

This paper focuses on a new session scheme and it's implementation in SmartSlog ADK [6], the code is available here http://sourceforge.net/projects/smartslog/. Goals of SmartSlog session are 1) manage many SSAP sessions in one session 2) operate in terms of such ontological

entities as individuals, their data and object properties 3) store and share data between SSAP sessions 4) handle subscriptions of different SSAP session and 5) cashing changes. SmartSlog session allows the developer to operate with many SSAP session as with one and it makes the application development easier. There are two versions of SmartSlog: ANSI C and .NET (C#). Currently the SmartSlog session is partially implemented and being tested in the C# version of ADK.

The rest of the paper is organized as follows. Section II describes the basic SSAP session and SmartSlog session. Section III analyzes a life cycle of different objects that the SmartSlog session contains. Finally, Section IV summarizes the paper.

## II. KP-SIB SESSION

To access smart space a KP starts a SSAP session with SIB. In the SSAP protocol a session means transactions between Join and Leave operations [7]. Join begins the session, when the session is active then it is possible to perform other operations: manipulate with triples and with subscriptions. Leave operation terminates the session and all information associated with the session on the SIB side will be removed and subscriptions will be ended. Low-level KPIs do not have additional functions to extend the capabilities of the SSAP session. If one KP uses multiple SSAP sessions, then the developer needs to manage SSAP sessions manually: store data and subscriptions for different SSAP sessions.

A session on the SmartSlog level includes a set of the SSAP sessions, it is possible to switch between them and control them. For example there are two smart spaces (see Fig. 1): one is in the house, and other is outside, some person has a KP and works with both smart spaces in a single session. When the person is inside the house, then one SSAP session will be active, when the person is outside, then other SSAP session will be active. Both two SSAP sessions work in one SmartSlog session and all data are available to use, regardless of current location of the person and active SSAP session.

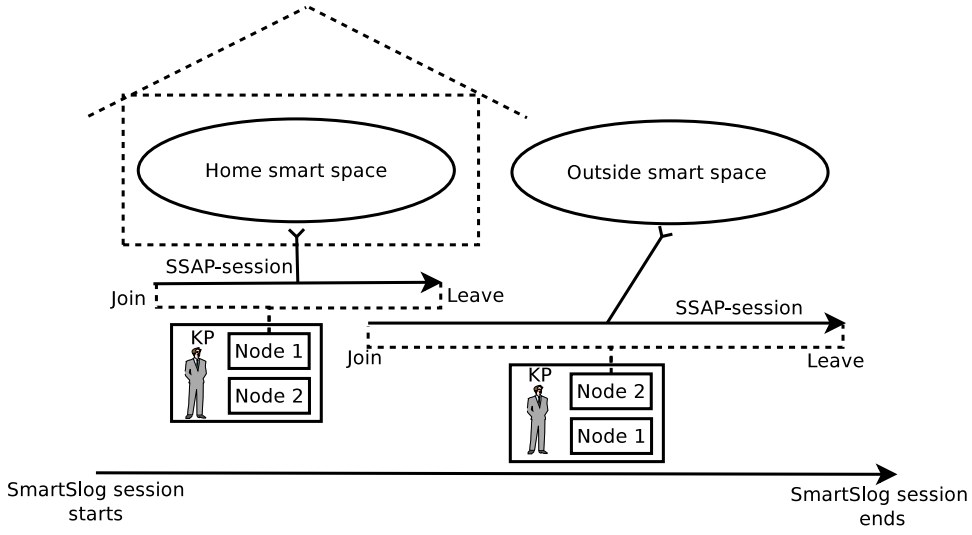


Fig. 1. SmartSlog session is based on several SSAP sessions

SmartSlog session includes a set of objects and different settings. Also it controls interaction between objects in the session. SmartSlog session includes (see Fig. 2): 1) set of nodes, 2) KPIs, 3) repository with ontology entities (classes, properties, individuals), 4) subscription information 5) properties changes 6) settings.

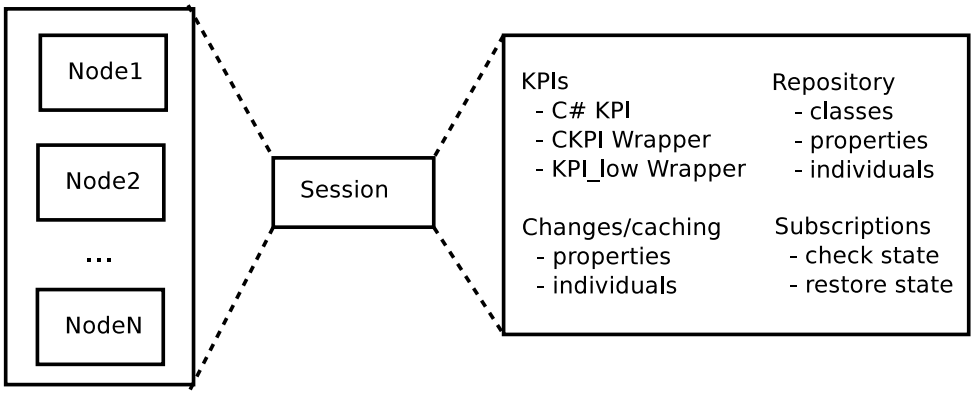


Fig. 2. SmartSlog session: components

The node is an object that represent a one SSAP session, but instead of using triples as do low-level KPIs it manipulates with ontology entities. The node does not store any data, except information about connection and SSAP session state, it only prepares data for low-level KPI and handles KPI's responses. Each node works in the SmartSlog session: in the manually created or in the default. The default SmartSlog session created automatically when created a first node without explicitly defined session. When the SmartSlog session is created it is need to perform two mandatory actions: 1) find available KPIs and 2) initialize local repository. If the KPIs are not found or the repository initialization is failed, then the further work is impossible: there is no access to the smart space without KPI and without the repository there is no way to store local data.

SmartSlog can work with different KPI through special KPI wrapper interface. KPI needs to implement this interface and then SmartSlog can load KPI in run-time and use it. The session finds and initialize KPI, then nodes can get an instance of KPI and work with it to communicate with the SIB.

Each session has a repository that is used as a store for ontology entities: classes, properties and individuals. The session repository is shared for all nodes in the session as Fig. 3 shows: 1 — one node adds new individuals to the repository and 2 — other nodes can work with these individuals. In this way it is possible to transfer individuals from one smart space to another or work with many smart spaces in one local space.

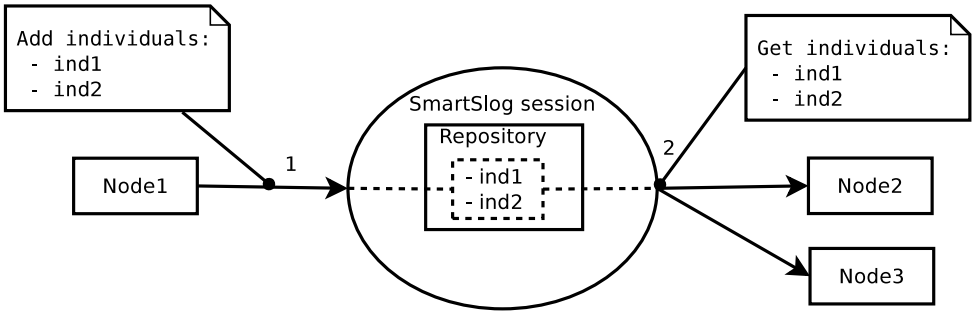


Fig. 3. SmartSlog session: repository

SmartSlog session controls access to different objects. Nodes can not directly access KPI or repository, session object provides necessary functionality for them or possibility to get needed object. Objects are provided through interfaces, it reduces coupling between objects

TABLE I
CODE EXAMPLES OF CREATING KP

| Code | Description |
|---|---|
| `ISession session = SessionManager.CreateSession(); Node node = new Node(session, "Node1", "X", "192.168.1.5", 10010);` | Create a session and a node. The node will be registered in the given session. If the session is not passed then the node will be registered in the default session. It is possible to get the default session and work with it as with manually created session. |
| `Property Lat = new Property(...); OntologyClass Point = new OntologyClass(...); session.RegisterEntity(Lat); session.RegisterEntity(Point);` | Create and register ontologies entities: class Point and property Location. You can use for operations only registered entities: create individuals based on classes and use properties to link individuals with object and data values. |
| `Individual point = node.CreateIndividual(Point); Individual lat = node.CreateIndividual(Lat); point.SetProperty(Lat, "0.763556"); node.Insert(point);` | After registration it is possible to work with ontology entities: create them, set properties for individuals and use entities for remote operations, in the example the individual is inserted to the smart space. Individuals are registered in the session automatically while creating. If session has many nodes then all nodes can get individuals or other entities, that were registered in the current session. |
| `SessionManager.DestroySession(session);` | Destroy the session. All nodes in the session unsubscribe from data and leave smart spaces. |

and allows to change inner logic without modifying client modules. For example, nodes do not have direct link to the repository and do not know how to store ontology entities in the session, but can request an object from the session or add an object to the session for the future usage. If the structure of the repository is changed or functionality is improved (for example, caching object to a file or database), then other objects stays without modifying.

## III. LIFECYCLE OF THE OBJECTS

All objects that are used in the SmartSlog belong to the session. The life cycle of an object starts when it is registered in the session and ends with the session termination or when object is removed from the session. In this case objects are nodes, ontology entities and subscriptions. A base template for the creating a KP contains four main steps (see Table I):

1) creating the node and the session;
2) registering objects that are represented by the ontology in the session (classes, properties);
3) manipulating with registered objects, interactions with the smart space, creating/registering individuals;
4) session termination.

Node automatically registers in the session (during the creation). After this it's life cycle begins and node has an access to needed data and objects in the session. The node can be removed from the session, in this case it's life cycle ends and it can not be used for working.

The ontology entities registration is needed to work with them and to maintain the integrity of the ontology. Registered classes determine what individuals can be created. Registered

properties determine what data or links can be set for individuals. The registration of ontology entities is performed by the developer. Usually classes and properties are used for the entire session and are registered at the beginning of the session. It is not necessary and needed entities can be registered at any time.

Unlike classes and properties, individuals are automatically registered by the SmartSlog. For example when a new object-property is received from the smart space. Such property refers to another individual, if this individual is in the session, then it will be used as a value for the property, otherwise the new individual will be created and registered. Removal of individuals should be performed by the developer. If the object-property is removed, then the SmartSlog removes property but not a value (individual). The individual will remain in the session. If a KP works with many individuals, then it is needed to remove individuals that are no longer needed from the session manually. When individual is removed, then it's life cycle is end and it can not be used for operations.

It is need to check relationship between ontology entities when some entity is removed from the session. The SmartSlog supports removing of: 1) class, if there are no individuals of this class in the session, 2) property when it is not set for some individual 3) individual, when other individuals are not linked with it and 4) individual with links (before removing individual all links to this individual will be deleted too).

Registering and removing of subscription are held by the SmartSlog automatically. When the data is subscribed the subscription will be registered, after unsubscription it will be removed.

## IV. Conclusion

This paper studied the session between KP and SIB operation. The SSAP session which is based on RDF triples and which is used in low-level Smart-M3 KPIs was analyzed and it's drawbacks were identified. Then a new SmartSlog session scheme was proposed and it expands and provides additional functionality as compared with the SSAP session.

The SmartSlog session scheme allows to combine many SSAP session in one session. The KP developer uses one session, but has many connections to different SIBs. All data that are received from SIB are stored in the session and shared between SSAP sessions. Using this scheme it is possible to transfer data from one SIB to another. It is also possible to work with one local space that actually consists from several smart spaces. This scheme now is under development and it is used in the SmartSlog ADK (C# version). The developer does not need to deal with many different SSAP sessions and does not need to manage them in Smart-M3 applications. The scheme can be enhanced with caching properties changes and managing subscriptions. This topic is left for further research.

## Acknowledgment

This research is a part of grant KA179 "Complex development of regional cooperation in the field of open ICT innovations" of Karelia ENPI programme, which is co-funded by the European Union, the Russian Federation and the Republic of Finland. The article was published with financial support from the Strategic Development Program of Petrozavodsk State University. I am grateful to the Open Innovations Association FRUCT for its support and R&D infrastructure. I would also like to thank Sergey I. Balandin and Iurii A. Bogoiavlenskii for their feedback and expertise.

## REFERENCES

[1] D. J. Cook and S. K. Das, "How smart are our environments? an updated look at the state of the art," *Pervasive and Mobile Computing*, vol. 3, no. 2, pp. 53–73, 2007.

[2] I. Oliver, "Information spaces as a basis for personalising the semantic web," in *Proc. 11th Int'l Conf. Enterprise Information Systems (ICEIS 2009)*, May 2009, pp. 179–184.

[3] J. Honkola, H. Laine, R. Brown, and O. Tyrkkö, "Smart-M3 information sharing platform," in *Proc. IEEE Symp. Computers and Communications*, ser. ISCC '10. IEEE Computer Society, Jun. 2010, pp. 1041–1046.

[4] S. Balandin and H. Waris, "Key properties in the development of smart spaces," in *Proc. 5th Int'l Conf. Universal Access in Human-Computer Interaction. Part II: Intelligent and Ubiquitous Interaction Environments (UAHCI '09)*. Springer-Verlag, 2009, pp. 3–12.

[5] D. G. Korzun, S. I. Balandin, V. Luukkala, P. Liuha, and A. V. Gurtov, "Overview of Smart-M3 principles for application development," in *Proc. Int'l Conf. Artificial Intelligence and Systems (AIS 2011)*, Sep. 2011.

[6] D. G. Korzun, A. A. Lomov, P. I. Vanag, J. Honkola, and S. I. Balandin, "Multilingual ontology library generator for Smart-M3 information sharing platform," *International Journal on Advances in Intelligent Systems*, vol. 4, no. 3&4, 2011.

[7] SOFIA: Smart Objects for Intelligent Applications, "Deliverable 5.22: Logical Service Architecture," 2011. [Online]. Available: http://www.sofia-project.eu/node/329.