



# ProMoNet Conceptual Solution Design for Dynamic Configuration Management of Networked Industrial Systems

Authors: Pekka Isto, Tommi Parkkila




Confidentiality: Public



<b>Report's title</b> ProMoNet Conceptual Solution Design for Dynamic Configuration Management of Networked Industrial Systems	
<b>Customer, contact person, address</b>	<b>Order reference</b>
<b>Project name</b> Product Life-Time Configuration Management of Networked Industrial Systems Verkotettujen teollisten järjestelmien elinkaarenaikainen konfiguraatiohallinta	<b>Project number/Short name</b> 73147 / ProMoNet
<b>Author(s)</b> Pekka Isto, Tommi Parkkila	<b>Pages</b> 23/75
<b>Keywords</b> Configuration management, dynamic, industrial systems	<b>Report identification code</b> VTT-R-04516-13
<p><b>Summary</b></p> <p>State-of-the-art high technology products are modular and multi-technical systems. Modular, model and component-based design methods boost system development, and give opportunity for more fine grained system management and maintenance than before. Data communications networks become more and more pervasive allowing products to have sophisticated networking capabilities and communication functions to send the required configuration management data to a server through Internet. These enablers make possible run-time and fine grained system management, e.g. remote system upgrade, extending product life-time by enabling system adaptation to face new set of requirements and standards in field.</p> <p>ProMoNet project defines and develops a conceptual solution for dynamic configuration management of networked industrial embedded systems and experimentally verifies the founded conceptual solution in a specific use case. The conceptual solution is aimed at providing configuration management functionality during the middle-of-life (MOL) phase, but also describes processes needed to transition a product data instance for the product from generic beginning-of-life product model to a configured and to be manufactured middle-of-life product model.</p> <p>The requirement elicitation process used is iterative. First, a literature survey and semi-structured industry interviews were done to construct the system context and preliminary requirements definition for dynamic configuration management. Second, the preliminary requirements were further refined with the knowledge gained from technology evaluations. At the third stage the requirements were presented to invited industry representatives for evaluating the value and ease of implementation for each individual requirement with an online survey. Based on the analysis of survey responses, 21 requirements were selected to key requirements set.</p> <p>The dynamic configuration management system was designed and modelled in OMG Systems Modelling Language using the SYSMOD methodology. SysML reuses a subset of UML 2 and provides extensions for modelling aspects other than software such as hardware, information, (continuous) processes, personnel, and facilities. SYSMOD is a methodology for systems engineering that produces SysML diagrams in a defined sequence using an iteratively incremental process. SysML and SYSMOD proved to be good choices for modelling language and methodology for the project. They are fairly easy and fast to pick up at least for someone familiar with model based design principles and UML in particular. The main shortcoming was the flatness of the domain knowledge diagram which does not capture the object oriented design principles that software engineers are used to rely on.</p> <p>The most critical part of the conceptual solution design was verified by implementing the functionality related to the scenario of local service reconfiguration. Parts of the demonstrator were implemented from the subset of the conceptual design pertinent to the scenario by different persons located at geographically different sites. There was very little need for</p>	

design refinement and integration effort other than at the implementation level. The successful demonstration shows that the critical part of the design is correct and at sufficient level of detail for implementation.

There are some noteworthy limitations in the conceptual solution design presented here. None of the security protocols other than the secure code were implemented in the demonstrator and none of them have been reviewed by information security experts. The conceptual solution design has not been validated with any particular business model. Further, the design is high-level conceptual and does not capture many important system level technological issues such as robustness, reliability and scalability.

<b>Confidentiality</b>	Public	
Oulu 20.6.2013 <b>Written by</b>  Pekka Isto, Senior Scientist	<b>Reviewed by</b>  Vesa Pentikäinen, Senior Scientist	<b>Accepted by</b>  Mikko Sallinen, Technology Manager
<b>VTT's contact address</b> Kaitoväylä 1, P.L. 1100, 90571 Oulu		
<b>Distribution (customer and VTT)</b> SEC of America, Sandvik Mining and Construction Oy, Wapice Oy, Microteam Oy, Miradore Oy, Tekes, VTT.		
<i>The use of the name of the VTT Technical Research Centre of Finland (VTT) in advertising or publication in part of this report is only permissible with written authorisation from the VTT Technical Research Centre of Finland.</i>		

## Preface

---

This research report presents the main results from research project Product Life-time Configuration Management of Networked Industrial Systems (Verkotettujen teollisten järjestelmien elinkaarenaikainen konfiguraatiohallinta). The jointly funded project started at the beginning of February, 2011, and ended at the end of April, 2013. The participating companies were SEC of America, Sandvik Mining and Construction Oy, Wapice Oy, Microteam Oy, and Miradore Oy. The steering group consisted of Tim Seaton, Severi Eerola, Pasi Tuominen, Pertti Arjanne, Mika Liukko and Mikko Sallinen (VTT). Tekes was represented by Martti Huolila. The authors want to thank the steering group members for their contributions and guidance during the project. The authors also want to thank the industry experts who were interviewed during the field study phase of the project for their time and valuable expertise as well as those who responded to the online survey. The financial support of Tekes, the companies, and VTT is also highly appreciated.

Oulu 20.6.2013

Authors

## Contents

---

Preface.....	3
Contents.....	4
1. Introduction.....	5
2. Requirements Elicitation Process.....	7
3. ProMoNet Conceptual Solution Design Overview .....	8
3.1 SysML and SYSMOD .....	8
3.2 System context and system process diagrams .....	9
3.3 Some key elements of the solution .....	13
3.4 A roadmap to the full Conceptual Solution Design .....	14
4. Verification and Limitations of the Model.....	14
5. Conclusions .....	15
6. Summary .....	16
References.....	17

## 1. Introduction

---

State-of-the-art high technology products are modular and multi-technical systems. There are several different parties involved in design, development, supply, and maintenance during the life-time of a product. These parties could include design houses, integrators, component suppliers, end users, and so on. End users have demanding requirements for product maintenance after the final installation. Product configurations depend on end users particular requirements, their financial potential, and available product components on the market. The particular combination of hardware and software components and features of the product evolve over its life-time as components become obsolete or better technology emerges for product upgrades. In industrial systems, after-sales markets are the fastest growing portions of products life-time chains. However, configuration management is most often processed offline, and products are often treated as single integrated systems. There is very little if any configuration management of products in active use in the field, and there are no mature methods or tools for such configuration management especially ones that support reconfigurable hardware such as Field-Programmable Gate Arrays.

Modular, model and component-based design methods boost system development, and give opportunity for more fine grained system management and maintenance than before. Data communications networks become more and more pervasive allowing products to have sophisticated networking capabilities and communication functions to send the required configuration management data to a server through Internet. These enablers make possible run-time and fine grained system management, e.g. remote system upgrade, extending product life-time by enabling system adaptation to face new set of requirements and standards in field. New technological solutions can be taken into use parallel to existing solutions by upgrading only required configuration modules. Fine grained remote configuration management sets new challenges for product configuration and product data management. These challenges have to be taken into account from the early design phase of a product already. Integrated and systematic management of product configuration over product's life-time is largely an unsolved issue with potential benefits for all stakeholders.

ProMoNet project provides solutions for life-time management and maintenance of networked modular, multi-technical products. Project target group consists of companies, which have a stake in the product during its life-time: product developers and manufacturers, design houses, component suppliers, and end users. ProMoNet defines and develops a conceptual solution for dynamic configuration management of networked industrial embedded systems and experimentally verifies the founded conceptual solution in a specific use case. The project is based on a three phase product life-cycle model, including beginning-of-life, middle-of-life, and end-of-life phases (Figure 1). The conceptual solution is aimed at providing configuration management functionality during the middle-of-life (MOL) phase, but also describes the requirements, functionality and processes needed to transition a product data instance for the product from generic beginning-of-life product model to a configured and to be manufactured middle-of-life product model.

The next chapter describes the requirements elicitation process used to capture the requirements for the conceptual solution using literature surveys, industry interviews and Wiegner's requirement prioritization method. The stakeholder descriptions and requirements are given in an appendix A. The following chapter describes the modelling language SysML and methodology SYSMOD used to construct ProMoNet Conceptual Solution Design for dynamic configuration management of networked industrial systems. The chapter presents the System Context and System Process diagrams for a high-level presentation of the

solution. It also discusses the key elements of the solution and gives guidance in studying the full model which is given as SysML diagrams in an appendix B. Chapter 4 describes the verification of the critical part of the conceptual solution design model with a demonstration implementation and discusses the limitations of the model and its verification. The final chapter states the conclusions.

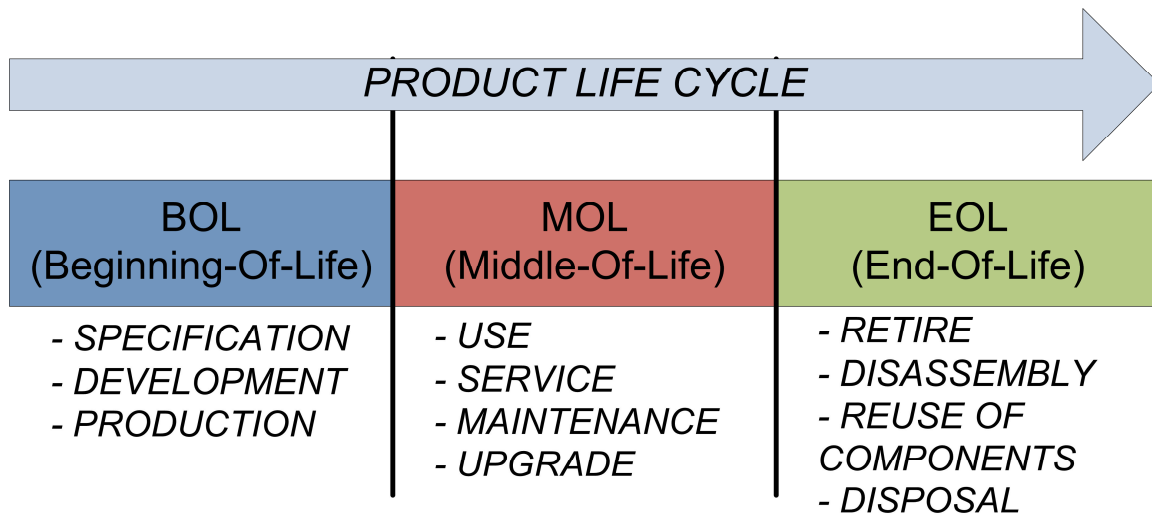


Figure 1. Product life-cycle model used in ProMoNet project.

Table 1. The Industry Interviewees.

Company name	# of Persons Interviewed	Category of Products	Role of Company	Applied Computing Technologies	System Architecture	Remote Connectivity
C1	8	Mechatronic Machines	OEM	Embedded PC , Microprocessor, PLC, CHW	Distributed	Yes
C2	2	Mechatronic Machines	OEM	Embedded PC , Microprocessor, PLC	Distributed	Yes
C3	1	Mechatronic Machines	OEM	MCU	Single	No
C4	1	Machine Condition Monitoring Systems	OEM	Embedded PC , Microprocessor	Single	Yes
C5	2	Machine Condition Monitoring Systems	OEM	Embedded PC , Microprocessor, CHW	Single	Yes
C6	1	Control Electronics	SUB	Embedded PC , Microprocessor	Both	Yes
C7	2	Control Electronics	SUB	Embedded PC , Microprocessor, PLC, CHW	Both	Yes



## 2. Requirements Elicitation Process

---

The requirement elicitation process used in ProMoNet is iterative and has three stages. First a literature survey and semi-structured industry interviews were done to construct the system context and preliminary requirements definition for three subareas of dynamic configuration management, namely middle-of-life Product Data Management systems, machine-to-machine (M2M) communication systems, and control electronics of the products. The interviews included 17 experts from seven organisations from the machine industry from Finland and USA (Table 1). The experts were interviewed about their organization in general, their product structures and features, configuration management processes applied on products, remote connectivity of products, and aftermarket services the products have during their middle-of-life phase.

The preliminary requirements were further refined with the knowledge gained from technology evaluations that focused on middle-of-life configuration management functionality provided by commercial Product Data Management systems, available short and long range wireless communication technologies, and available M2M communication platforms. The middle-of-life PDM system evaluation started with literature survey to find existing middle-of-life PDM system categories. A representative system from each category recognized was selected for deeper study and a semi-structured interview of a company representative of the system's provider was conducted. Three interviews were conducted and analysed, and the findings were documented and published elsewhere [Parkkila et al. 2012].

For the wireless communication technology evaluation a literature survey was performed to find a wide range of wireless communication technologies, their categories and classifications. Pre-screened short and long range wireless communication technologies were evaluated against the preliminary requirements which were then further developed to explicate implicit requirements that eliminated some of the technologies from the list of suitable ones.

The preliminary requirements were the least developed in the M2M communication system area and not really rigorous enough to make an informed decision about the M2M technology platform needed for an implemented system. However, a decision was made to not further elaborate the preliminary requirements in this area since they can be augmented with the requirements from the ETSI TS 102 689 M2M Service Requirements technical standard [ETSI TS 102 689, 2010].

After the refinement of the preliminary requirements through technology evaluations and merging overlapping requirements, the set of requirements included total of 40 requirements (Appendix A). The requirements were presented to invited industry representatives for evaluating the value and ease of implementation for each individual requirement with an online survey. The online survey was based on Wiegner's requirement prioritization method [Wiegner, 2003]. Total of 20 respondents were invited and eleven completed surveys were received, three from product managers, six from R&D personnel, one from production personnel, and one from aftermarket personnel. The ranking results from the survey were analysed with and without aftermarket bias to find requirements with most stakeholder value. Based on the analysis, 21 requirements were selected to key requirements set.

The conceptual solution design is based on the key requirements and the requirements rephrased for brevity and referenced to the original requirements are included in the model (Appendix B).

### 3. ProMoNet Conceptual Solution Design Overview

---

#### 3.1 SysML and SYSMOD

In order to capture the system engineering aspects of the dynamic configuration management system, it was decided to be modelled in OMG Systems Modelling Language [OMG 2012] using the SYSMOD methodology [Weilkiens 2008]. OMG Systems Modelling Language (SysML) originates in the International Council on Systems Engineering's (INCOSE) decision to adapt UML to systems engineering applications. SysML reuses a subset of UML 2 and provides extensions for modelling aspects other than software such as hardware, information, (continuous) processes, personnel, and facilities. Noteworthy, SysML includes requirements and parametric relationships as a distinct diagram types and lessens the object orientation of UML to a point where the concepts of object oriented design can be completely absent in a SysML model. SysML retains the structural and behavioural diagram types of UML 2 but some are extended to better suite the needs of systems engineering and renamed.

SYSMOD is a methodology for systems engineering that produces SysML diagrams in a defined sequence. The SYSMOD approach starts with describing the project context that is the goals for the system, its environment and situation and preliminary ideas for realizing the system. This step does not produce any SysML diagrams but rather a text document along the lines of the introduction to this report. The first diagrams emerge from the second step of identifying the system stakeholders, collecting the requirements, and representing them as requirements diagram. System context diagram describes the system with in its environment including actors and external systems interacting with it by information flows to some interaction points. The services that the system provides are modelled with use case diagrams and essential step descriptions. Use case modelling is a step when modelling of the system's information elements in a domain knowledge diagram should be started. The logical flow dependencies between the use cases are described in a system process diagram. Paths through the system process diagram describe scenarios of system usage. Use case diagrams are refined to use case flow diagrams which describe the activities of the use case and the flows between them and further to use case object flow diagrams that model also the objects that flow between activities. SYSMOD continues with steps to describe how the use cases are realized. These steps produce diagrams that describe system's interaction with actors, interfaces, system's internal structures and their state models.

Although the SYSMOD approach is described as a linear process above, it is actually an iteratively incremental process where the model is constructed in slices and the existing parts of the model are modified as deeper understanding of the system as a whole emerges during the modelling. The approach can also be tailored to the needs of the specific project. In this particular project the model omits interfaces, and interaction and state diagrams as those were deemed non-essential.

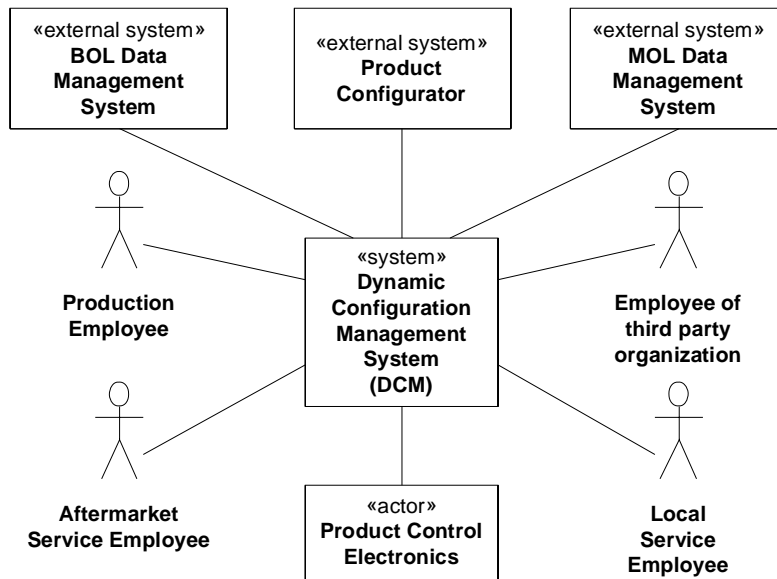


Figure 2. System Context diagram.

### 3.2 System context and system process diagrams

Figure 2 presents the system context of the Dynamic Configuration Management System (DCM) that is the system to be modelled here. The DCM connects to three external systems. BOL Data Management System is the product data management software and platform for storing beginning-of-life (as-designed) product data. Product Configurator is data management software and platform for specifying and generating product configuration data structures according to customer requirements. MOL Data Management System is data management software and platform for storing evolving product unit specific middle-of-life (as-built and as-maintained) data structures. The external systems work together such that Product Configurator is used to instantiate the product unit specific data structure from the generic product design data in the BOL Data Management System and this data structure is stored and maintained in MOL Data Management System.

The actors in Figure 2 have different roles in the product's life-cycle. Production Employee produces customized instances for product's end users, i.e. creates individual product unit data structures in MOL Data Management System. Production Employee needs always the most recent versions of configuration items and their descriptions for new products to be produced. Production Employee also wants to know when new features are available.

Aftermarket Service Employee is the primary stakeholder of the DCM system and interested in offering as flexible and effective after sales maintenance service for deployed products as possible. Aftermarket Service Employee is also interested in remote online configuration management of networked products or tasking Local Service Employees in the field to perform reconfiguration locally if the product is not connected to the network. Aftermarket Service Employee wants to get information about the usage of the product, maintenance history, faults, status of the product, and diagnostics to find aftersales opportunities related to the deployed products.

Local Service Employee gets reconfiguration tasks for deployed products from the DCM system and performs the reconfigurations locally through local wireless or wired communication interface to the product. Local Service Employee also retrieves information about the usage of the product, maintenance history, faults, status of the product, and diagnostics, and uploads the data to MOL Data Management System once again within network coverage.

Employee of third party organization is granted restricted access to DCM system to perform specific tasks such as obtaining limited configuration and diagnostic data from the MOL Data Management System for a specific product unit to investigate fault in a subsystem of the product that is subcontracted to the third party organization.

Figure 3 gives the top level system process diagram for the DCM system. Figure 4 expands the Change Product configuration system process of Figure 3. Together these diagrams describe the service scenarios of the system. For a specific product instance to be managed by the DCM system it needs to have its product data brought from BOL Data Management System to MOL Data Management System in use case Create Product Specific Data Structure and have its original factory configuration data from Product Configurator stored to MOL Data Management System. The current configuration of the product stored to MOL Data Management System can be accessed by the Aftermarket Service Employee (Read Current Configuration from PLM) and Local Service Employee (Retrieve Current Configuration from PLM). The active configuration of a product in the field can be accessed over the network by the Aftermarket Service Employee (Pull Current Configuration from Product) and over the local communication interface by Local Service Employee (Get Current Configuration from Product). Retrieve Service Code is a use case that grants access to a particular product in the field to a particular Local Service Employee. The use case is elaborated in the next subchapter.

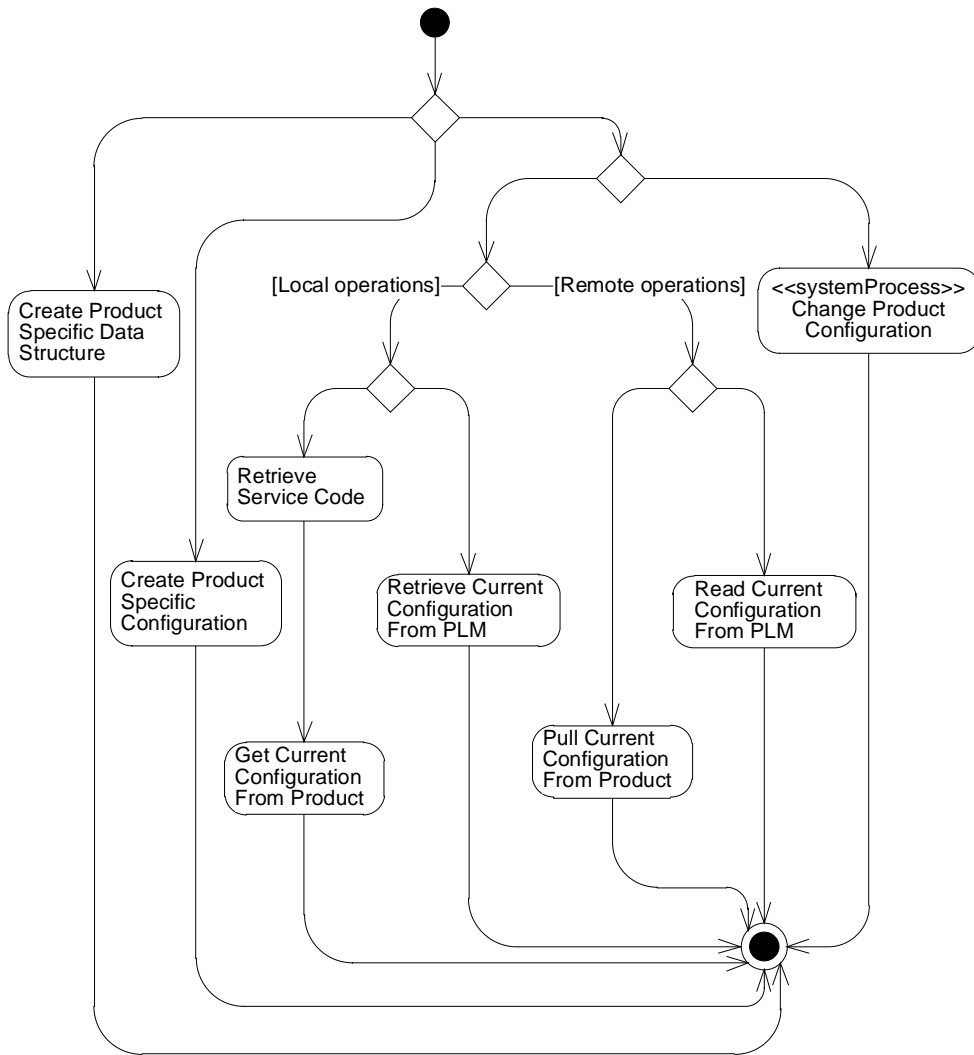


Figure 3. Top-Level System Process diagram.

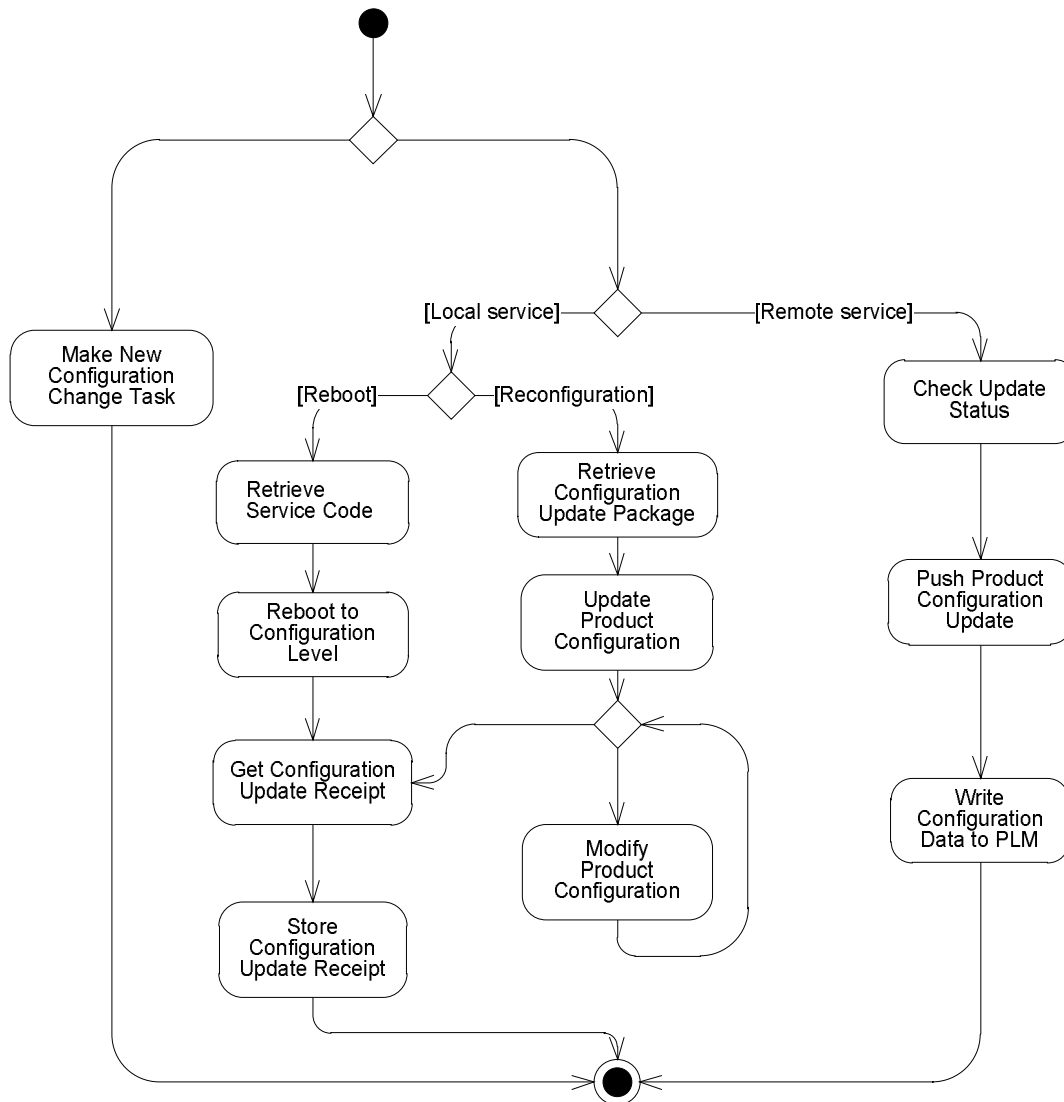


Figure 4. System Process diagram for Change Product Configuration.

For a product in the field to be reconfigured Aftermarket Service Employee must first create a reconfiguration task in use case Make New Configuration Change Task (Figure 4). If the reconfiguration can be safely performed remotely over the network, Aftermarket Service Employee checks the availability of a configuration update for a particular product (Check Update Status), makes the reconfiguration available to the product to perform (Push Product Configuration Update), and, after the product reconfigures itself, stores the updated configuration from the product to MOL Data Management System (Write Configuration Data to PLM). Often the product to be reconfigured is not within network coverage or the final configuration requires tuning in the field. In such cases the reconfiguration is performed locally by the Local Service Employee who downloads the configuration update from MOL Data Management to a Service Terminal when under network coverage (Retrieve Configuration Update Package). Local Service Employee connects to the product using the local communication interface and transfers the configuration update from the Service Terminal to the product (Update Product Configuration), possibly modifies the configuration (Modify Product Configuration), and retrieves the final configuration from the product to the Service Terminal (Get Configuration Update Receipt). Once again within network coverage, Local Service Employee transfers the final product configuration to MOL Data Management System (Store Configuration Update Receipt).

In the case of system malfunctions or reconfiguration errors the product on the field must be recovered to a safe configuration. Each product stores three configurations: The original factory configuration, last operational configuration and the current configuration. Once the access to the product is obtained (Retrieve Service Code), Local Service Employee can reboot the product to any of those three configurations (Reboot to Configuration Level) and store the final configuration to MOL Data Management System as in the reconfiguration scenario.

### 3.3 Some key elements of the solution

There are a few noteworthy features in the conceptual solution design. It's a key requirement that the access to the products in the field is controlled with identification, authentication and authorization mechanisms. However, due to possible sporadic network access to the products, long service life of the products considered in this project, and limited amount of storage in the product control electronics is it infeasible to store a user database in the products that the actors in the system could be authorized against. The solution is to have each product guard access to itself with an one-time use secret key (secure code) which is transferred between the product and MOL Data Management System in encrypted form. The actors of the system are identified, authenticated and authorized against a user database in the MOL Data Management System and if access is granted the actor is given the secret key to the product. At the end of each operation the product generates new secret key and provides it to the actor in encrypted form to be uploaded to the MOL Data Management System for future operations.

While dynamic configuration management was partially inspired by the availability of inexpensive wireless communication devices and networks, the solution also provides functionality for off-line configuration management such that the product can remain outside of network coverage for longer times or even permanently. The required communication is performed by Local Service Employee with a portable computer or memory element used to transfer data to and from the product.

The configuration model described in the domain knowledge diagram fulfils the requirements coming from the field study. However, the configuration update data is opaque to the DCM system and the conceptual solution would also be workable with different configuration models. The configuration model actually becomes significant only in the internal functionality of the Product Configurator, MOL Data Management System, and product control electronics.

The design makes minimal commitments to any actual ICT architecture that would be used to realize the conceptual solution. The internal structure of the DCM system (see Appendix B) is such that the internal blocks of the system can be deployed in a multitude of ways to different devices and product control electronics elements. At the minimum there has to be some mobile storage element such as a USB memory stick that is used to transfer the required configuration related data from the MOL Data Management System to the product control electronics with all the other functionality except Product Services deployed to a central server with the MOL Data Management System. At the other extreme, should the product have reliable network connection and considerable computational resources, most of the internal blocks of the system could be deployed on the product control electronics with the exception of the Product Configurator and BOL Data Management System which obviously would be owned by sales and engineering functions of the manufacturer. The most plausible deployment architecture would have Product Life-cycle Data Management System, DCM Mobile Communications and Data Security and Access control Service deployed on server(s), Terminal deployed on office PC(s), mobile Terminal deployed on mobile devices such as laptops or tablet computers, and Product configuration Managements System deployed on the product control electronics.



The use case flow diagrams have a recurring SysML design pattern which has the arrival of a signal to generate an object of the same name. This is a modelling decision that originates from the property of SysML that top-level activity diagrams do not have parameters. An actual software implementation of the activity would likely have it the other way around. The arrival of software message or event object would signal the pending activity to start.

### 3.4 A roadmap to the full Conceptual Solution Design

The further study of the conceptual solution design is best continued by reading the use case diagrams and the narratives coming with them. The use case diagrams alone describe the structure of the use case and how information flows between the parts of the decomposed use case, and the actors and external systems related to the use case. The essence description gives the order in which the parts of the use case execute. The domain knowledge diagram describes the information elements – blocks in SysML and objects in UML – that flow in the use case. The details of the activities and the object flows between activities and the environment can be found in use case flow diagrams.

Once the behavioural aspects of the DCM system are familiar, the structural aspects can be studied starting with the external interaction ports of the system. Internal structure and the communication ports of the internal blocks are modelled explicitly but the object flows between internal blocks are only available indirectly by mapping the relevant activities in the use case flow diagrams to internal blocks and observing which objects flow between activities mapping to different internal blocks.

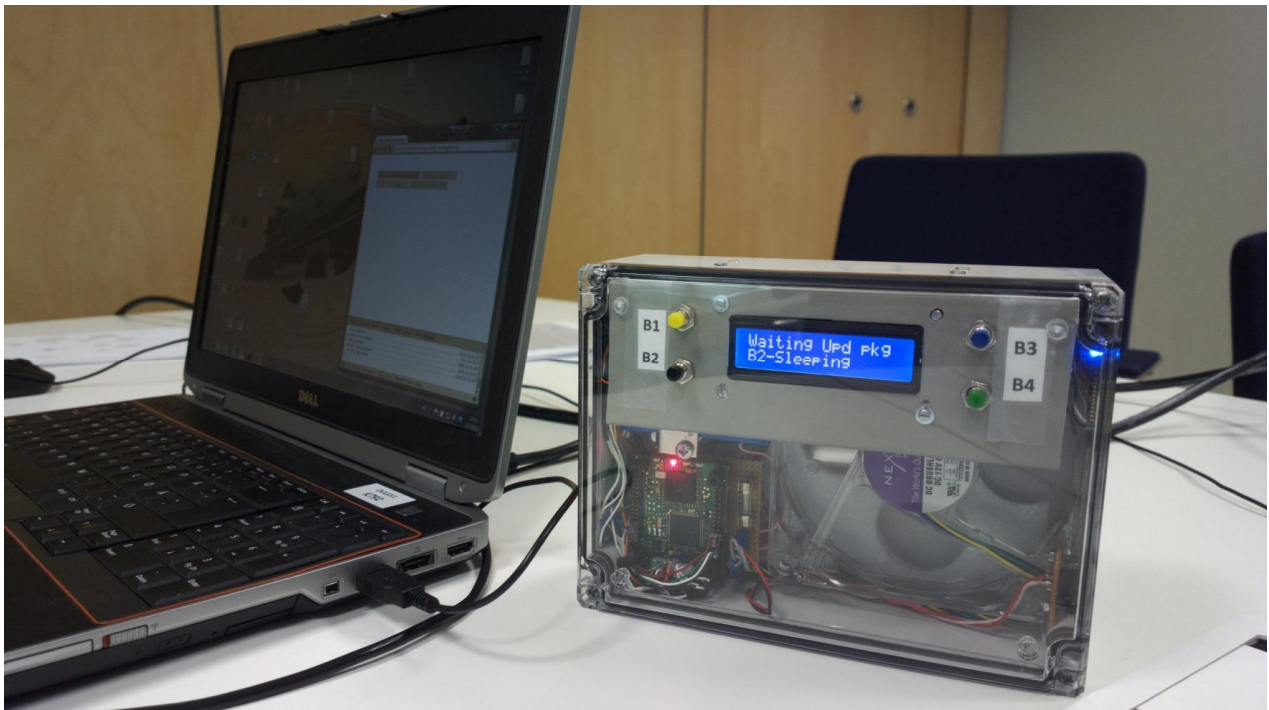


Figure 5. The demonstration device and Service Terminal User Interface on a laptop.

## 4. Verification and Limitations of the Model

---

The most critical part of the conceptual solution design was verified by implementing the functionality related to the scenario of local service reconfiguration performed by Local Service Employee (see Figure 4).



The demonstration product is an advanced fan (Figure 5) for which the user has originally purchased the most basic configuration but has later ordered an oscillating fan upgrade from the aftersales. Aftermarket Service Employee has enabled the feature in the product unit specific data structure in MOL Data Management System. In the concrete scenario Local Service Employee connects to the fan with Bluetooth connection, downloads the new configuration from MOL Data Management System to a laptop, updates product configuration, fine-tunes the oscillation amplitude to suit the user, and gets the final configuration from the fan to the laptop to be uploaded to the MOL Data Management System at later time.

The demonstration product has been implemented with VTT Node wireless sensor node, and the limited MOL Data Management System emulator with MySQL, Python, and various web technologies. Both parts of the demonstrator were implemented from the subset of the conceptual design pertinent to the scenario by different persons located at geographically different sites. There was very little need for design refinement and integration effort other than at the implementation level. The successful demonstration shows that the critical part of the design is correct and at sufficient level of detail for implementation.

There are some substantial limitations in the conceptual solution design presented here. None of the security protocols other than the secure code were implemented in the demonstrator and none of them have been reviewed by information security experts. The conceptual solution design has not been validated with any particular business model that would describe how added value is generated by the DCM system and how the value is distributed in the value chain related to it. Further, the design is high-level conceptual and does not capture many important system level technological issues such as robustness, reliability and scalability. The demonstrator clearly solves only a “toy problem” and while it verifies procedural and modelling aspects of the conceptual solution design, it leaves information technology system level aspects open.

## 5. Conclusions

---

The conceptual solution design for dynamic configuration management of networked industrial embedded systems was completed successfully and the critical part of it experimentally verified to be correct and at sufficient level of detail for implementation. The requirement analysis produced the key requirements with good coverage of middle-of-life Product Data Management and product control electronics aspects, but communication aspects are less developed. There are few system level requirements and this is reflected in the conceptual solution design.

SysML and SYSMOD proved to be good choices for modelling language and methodology for the project. They are fairly easy and fast to pick up at least for someone familiar with model based design principles and UML in particular. The main shortcoming was the flatness of the domain knowledge diagram which does not capture the object oriented design principles that software engineers are used to rely on. SysML language does not require the use of those principles and SYSMOD methodology does not encourage it. However, they do not preclude or discourage it either, and it's possible to combine SysML and UML. The diagrams were drawn with Visio using a stencil for SysML. Given the amount of modelling involved in the project, the use of a SysML modeller rather than generic drawing software would have made the work more efficient.

## 6. Summary

---

*State-of-the-art high technology products are modular and multi-technical systems. There are several different parties involved in design, development, supply, and maintenance during the life-time of a product. These parties could include design houses, integrators, component suppliers, end users, and so on. Modular, model and component-based design methods boost system development, and give opportunity for more fine grained system management and maintenance than before. Data communications networks become more and more pervasive allowing products to have sophisticated networking capabilities and communication functions to send the required configuration management data to a server through Internet. These enablers make possible run-time and fine grained system management, e.g. remote system upgrade, extending product life-time by enabling system adaptation to face new set of requirements and standards in field.*

*ProMoNet project defines and develops a conceptual solution for dynamic configuration management of networked industrial embedded systems and experimentally verifies the founded conceptual solution in a specific use case. The conceptual solution is aimed at providing configuration management functionality during the middle-of-life (MOL) phase, but also describes processes needed to transition a product data instance for the product from generic beginning-of-life product model to a configured and to be manufactured middle-of-life product model.*

*The requirement elicitation process used is iterative and has three stages. First, a literature survey and semi-structured industry interviews were done to construct the system context and preliminary requirements definition for dynamic configuration management. Second, the preliminary requirements were further refined with the knowledge gained from technology evaluations. At the third stage the requirements were presented to invited industry representatives for evaluating the value and ease of implementation for each individual requirement with an online survey. Based on the analysis of survey responses, 21 requirements were selected to key requirements set.*

*The dynamic configuration management system was designed and modelled in OMG Systems Modelling Language using the SYSMOD methodology. SysML reuses a subset of UML 2 and provides extensions for modelling aspects other than software such as hardware, information, (continuous) processes, personnel, and facilities. SYSMOD is a methodology for systems engineering that produces SysML diagrams in a defined sequence using an iteratively incremental process. SysML and SYSMOD proved to be good choices for modelling language and methodology for the project. They are fairly easy and fast to pick up at least for someone familiar with model based design principles and UML in particular. The main shortcoming was the flatness of the domain knowledge diagram which does not capture the object oriented design principles that software engineers are used to rely on.*

*The most critical part of the conceptual solution design was verified by implementing the functionality related to the scenario of local service reconfiguration. Parts of the demonstrator were implemented from the subset of the conceptual design pertinent to the scenario by different persons located at geographically different sites. There was very little need for design refinement and integration effort other than at the implementation level. The successful demonstration shows that the critical part of the design is correct and at sufficient level of detail for implementation.*

*There are some substantial limitations in the conceptual solution design presented here. None of the security protocols other than the secure code were implemented in the demonstrator and none of them have been reviewed by information security experts. The conceptual solution design has not been validated with any particular business model. Further, the design is high-level conceptual and does not capture many important system level technological issues such as robustness, reliability and scalability.*

## References

---

- ETSI TS 102 689: 2010. Machine-to-Machine communications (M2M); M2M Service Requirements, European Telecommunications Standards Institute. [http://www.etsi.org/deliver/etsi\\_ts/102600\\_102699/102689/01.01.01\\_60/ts\\_102689v010101p.pdf](http://www.etsi.org/deliver/etsi_ts/102600_102699/102689/01.01.01_60/ts_102689v010101p.pdf) (accessed 30 April, 2013)
- OMG: 2012. OMG Systems Modeling Language (OMG SysML™), Object Management Group, Inc. <http://www.omg.org/spec/SysML/1.3/PDF/> (accessed 30 April, 2013)
- Parkkila, T., Kääriäinen, J., Tanner, H., & Riekkilä, J., 2012. Middle-of-life PLM Solutions for Reconfigurable Networked Mechatronic Products. ARPN Journal of Systems and Software 2(5), pp. 177-186. [http://scientific-journals.org/journalofsystemsandssoftware/archive/vol2no5/vol2no5\\_3.pdf](http://scientific-journals.org/journalofsystemsandssoftware/archive/vol2no5/vol2no5_3.pdf) (accessed 30 April, 2013)
- Weilkiens, T. 2008. Systems Engineering with SysML/UML: Modelling, Analysis, Design. Burlington, MA, USA: Morgan Kaufmann OMG Press. 307 p.
- Wieggers, K.E. 2003. Software Requirements. Redmont, WA, USA: Microsoft Press. 544 p.

## Appendix A: Stakeholders and Requirements

Table 2. Stakeholders for dynamic configuration management system.

Stakeholder	Priority (1-4)	Comments/Interests
Product OEM, R&D	2	Develops products and new features following the set product data structure. Wants a view of product usage info, failures, executed maintenance, service events. Interested in integrating third party products R&D results and product descriptions into product data structure as well.
Product OEM, Production	2	Produces customized instances for end users, i.e. creates individual product data structures. Gets always the most recent versions of configuration items and their descriptions for new products to be produced. Wants to know when new features are available.
Product OEM, After Market	1	Interested in offering as flexible and effective after sales maintenance service for deployed products as possible. Interested in remote configuration management for networked products, interested in having a possibility to order a development task for some existing feature or a new feature from a third party after sales through the system. Eager for safe and secure methods for product configuration updates through online or offline channels. Wants to get information of usage of the system, maintenance history, faults, status of the product, diagnostics.
Subcontractor, R&D	2	Interested in seamless integration into system for uploading ordered, developed and tested sub-products to product data structure.
Subcontractor, After Market	1	Interested in seamless integration into system for monitoring usage of the provided sub-products and related faults and required bug fixes. Ability to upload updated sub-product data to deployed product data structure.
Third Party, Service	2	Interested in interfacing to deployed products' configuration data seamlessly through products' local communication interfaces and to have access to execute required configuration data modifications for deployed product on field. Wants to get information of usage of the system, maintenance history, faults, status of the product, diagnostics. Eagers for safe and secure methods for product configuration data download and upload to external information system.
End User	2	Interested in easy product maintenance. Need to have rights and easy way to choose and set reasoning which kind of configuration updates are possible to execute for the product remotely or locally. Updating can't threat safety against product or product's environment.
Tele Operators	4	Advices and specifies how the communication medias can be used to transmit data from information systems to machine and vice versa. Offers media for remote configuration management in cellular mobile phone network or through satellite network.
Data Service Provider	3	Interested in managing product data and deployed product data instances. Specifies interfaces how OEM parties and third party contractors can interface to data structures.
M2M Service Provider	3	Utilizes cellular and satellite networks for transmitting messages between peers of the system. Specifies also interfaces how deployed product can be interconnected to the system, and how information systems can be interconnected.

Table 3. Requirements for dynamic configuration management system.

#	Id	Requirement
<b>Req.1. Product Data Management</b>		
1	Req.1.1.	Every deployed smart product on field has own updateable product unit specific (as-built) data structure at their middle of life (MOL) cycle phase for product service and maintenance support.
2	Req.1.2.	The product unit specific data structure of a smart product is generated (semi-) automatically from the product's beginning of life (BOL) phase data management systems (e.g. ERP, PDM, PLM) using customer order specifications.
3	Req.1.3.	The product unit specific data structure of a smart product in the MOL phase data management system consists of all product HW and SW design data (or links to actual baseline documents), BOM, configuration items, configuration parameters, spare-parts, service manuals, service log files and reports, diagnostic data files, information of the application and operating environment of the product.
4	Req.1.4.	The product unit specific data structure of a smart product is stored into a central repository (MOL Data Management System) managed by OEM.
5	Req.1.5.	However, a part of the product unit specific data structure (e.g. a configuration data of a sub-part or sub-module) of a smart product can be physically stored into a separated data repository managed by OEM or a third party subcontractor, but the information is accessible through the main data structure.
6	Req.1.6.	Deployed smart product stores locally a copy of the product unit specific data structure with the configuration data of the product.
7	Req.1.7.	The product unit specific data structure of a smart product in the MOL phase data management system can be updated with a new data from the BOL phase data management systems. E.g. data of a new subsystem can be added into the unit specific data structure or existing parameter values can be replaced with new values.
8	Req.1.8.	The product unit specific data structure of a smart product in the MOL phase data management system can be updated by OEM aftermarket. E.g. OEM aftermarket can add new technical service bulletin, or change configuration parameter values.
9	Req.1.9.	The product unit specific data structure of a smart product described in the MOL phase data management system can be updated by the networked smart product itself.
10	Req.1.10	Every change or data update done to the product unit specific data structure of a smart product is stored in a service log file and signed with a meta information of the date of change, change location (local or information system), a party who executed the change.
11	Req.1.11	The product unit specific data structure stores at least three levels of configuration data history of configuration items as back-up data: factory configuration, previous configuration and the current configuration.
<b>Req.2. Multi-Organizational Access</b>		
12	Req.2.1.	Multiple organizations (e.g. OEM parties, subcontractors, third party service providers) can access the product unit specific data structure of a smart product in a collaborative environment.
13	Req.2.2.	Product OEM can give restricted access to other parties and organizations for the product unit specific data structure. As an example, there is a possibility to limit the visibility of product structure and related data to a specific sub-system only via user rights management (e.g. access rights to sub-system).
14	Req.2.3.	In addition to OEM parties, sub-contractors and third party service providers are able to execute updates to the product unit specific data structure of a smart product securely and remotely in a collaborative environment.
<b>Req.3. Machine-to-Machine offline communication</b>		
15	Req.3.1.	Product local service can download product configuration data of the product unit



		specific data structure of a smart product remotely from the MOL phase data management system to a mobile terminal (e.g. smart phone, tablet computer, laptop, proprietary mobile terminal, mass memory device) through mobile communication network (e.g. 3G, WiFi, Satellite), and update the configuration data on the smart product through short range wireless communication (e.g. Bluetooth, WiFi, Zigbee) or through stationary communication bus technology and protocol (e.g. USB, CAN, CANOpen, Profibus, Ethernet) applied in the product.
16	Req.3.2.	Product local service can download product configuration data and product diagnostic data of a smart product to a mobile terminal (e.g. smart phone, tablet computer, laptop, proprietary mobile terminal, mass memory device) through short range wireless communication (e.g. Bluetooth, WiFi, Zigbee) or through stationary communication bus technology and protocol (e.g. USB, CAN, CANOpen, Profibus, Ethernet) applied in the product, and update the data to the product unit specific data structure of the smart product in the MOL phase data management system remotely through mobile communication (e.g. 3G, WiFi, Satellite).
17	Req.3.3.	Product configuration data, diagnostic data, and management function calls can be exchanged between the products through short range wireless communication (e.g. bluetooth, WiFi, Zigbee). As an example, product configuration data upload from a mobile terminal can be delivered to the desired smart product through other smart products on field (Machine-to-Machine communication).
18	Req.3.4.	Every service or maintenance event executed for a smart product through any terminal is confirmed to the product unit specific data structure of the smart product in the MOL phase data management system. As an example, products confirm executed service events to terminal with a unique token (product specific and time and service related), which is transmitted to MOL phase data management system in order to get the service checked.
<b>Req.4. Machine-to-Machine online communication</b>		
19	Req.4.1.	Smart product can download up-to-date product configuration data from MOL phase data management system through mobile communication network (3G, Satellite).
20	Req.4.2.	Smart product can upload its configuration data and diagnostic data to MOL phase data management system through mobile communication network (3G, Satellite) and execute update for the product unit specific data structure of the smart product.
<b>Req.5. Smart Product</b>		
21	Req.5.1.	Smart product sends its manufacturing info (product id, manufacturer name, device model, OEM name, etc.) of all units of the system when requested by aftermarket service utilizing remote online connection or local connection via mobile terminal, or a stationary user interface unit of the product.
22	Req.5.2.	Smart product sends detailed information and status of its configuration (e.g. list of connected devices, version information of configuration items, status of all manageable options and functions, values of manageable parameters, service log files, special characteristic of memory size, CPU, CPU freq., etc.) when requested by aftermarket service utilizing remote online connection or local connection via mobile terminal, or a stationary user interface unit of the product.
23	Req.5.3.	Local aftermarket service can activate or de-activate specific configuration items (e.g. options) of smart products (4th level of CM hierarchy) utilizing local data connection via mobile terminal, or stationary user interface unit of the products.
24	Req.5.4.	Remote aftermarket service can activate or de-activate specific configuration items (e.g. options) of smart products (4th level of CM hierarchy) utilizing remote online connection.
25	Req.5.5.	Local aftermarket service can change values of run-time and boot-time application parameters of smart products (4th level of CM hierarchy) utilizing local data connection via mobile terminal, or stationary user interface unit of the products
26	Req.5.6.	Remote aftermarket service can change values of run-time and boot-time application parameters of smart products (4th level of CM hierarchy) utilizing remote online connection

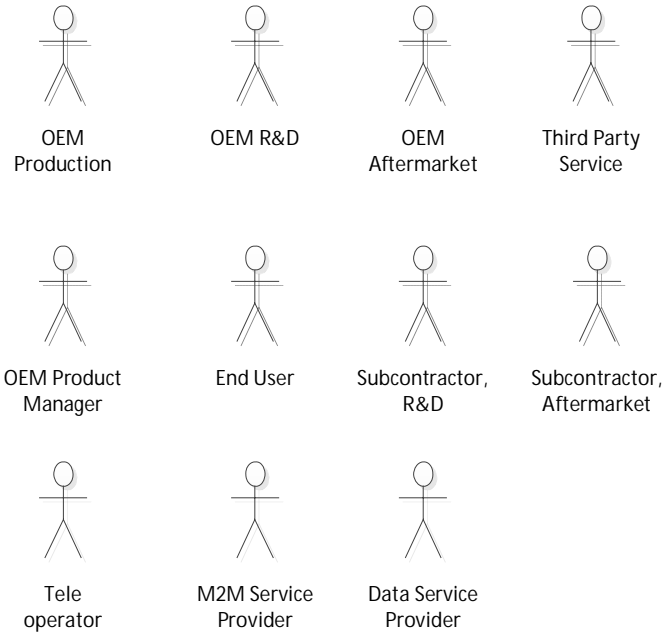
27	Req.5.7.	Local aftermarket service can execute software updates for smart products (3rd level of CM hierarchy) utilizing local data connection via mobile terminal, or stationary user interface unit of the products.
28	Req.5.8.	Remote aftermarket service can execute software updates for smart products (3rd level of CM hierarchy) utilizing remote online connection.
29	Req.5.9.	Local aftermarket service can execute firmware image updates for smart products and hardware design image updates for configurable hardware (e.g. Field Programmable Gate Arrays - FPGAs) on smart products utilizing local data connection via mobile terminal, or stationary user interface unit of the products. (2nd level of CM hierarchy).
30	Req.5.10	Remote aftermarket service can execute firmware image updates for smart products and hardware design image updates for configurable hardware (e.g. Field Programmable Gate Arrays - FPGAs) on smart products utilizing remote online connection. (2nd level of CM hierarchy)
31	Req.5.11	Local aftermarket service can reboot smart products to specific configuration history level: factory reset configuration, previous configuration, the newest configuration, utilizing local data connection via mobile terminal, or stationary user interface unit of the products.
32	Req.5.12	Remote aftermarket service can reboot smart products to specific configuration history level: factory reset configuration, previous configuration, the newest configuration, utilizing remote online connection.
33	Req.5.13	End user can restrict which management functions are allowed for the smart product through online (remote) or offline (local) connection.
34	Req.5.14	Smart product unit can execute auto configuration if new configuration data is available locally on a master unit or remotely in the MOL phase data management system.
35	Req.5.15	Smart product controls which local or remote management functions are allowed at different application run-times.
36	Req.5.16	Smart product does self-diagnosis and self-tests for its computing units and software; as an example it monitors general run-time parameters, memory status, CPU loads, communication channels' metrics, and the set diagnostic trap events, and collects log files of historical and/or statistical data of system performance, faults, application usage, and other diagnostic data.
37	Req.5.17	In a case of a serious system fault captured by the self-test function, smart product does automatic system reconfiguration to the previous working configuration.
38	Req.5.18	Smart product collects log files of all executed configuration management and service actions, with metadata of who has executed the action and when.
<b>Req.6. Mobile Communication</b>		
39	Req.6.1.	Communication between systems is safe and secure
40	Req.6.2.	Communication systems utilize standard technologies, which are commercially available

## **Appendix B: ProMoNet Conceptual Solution Design Model**

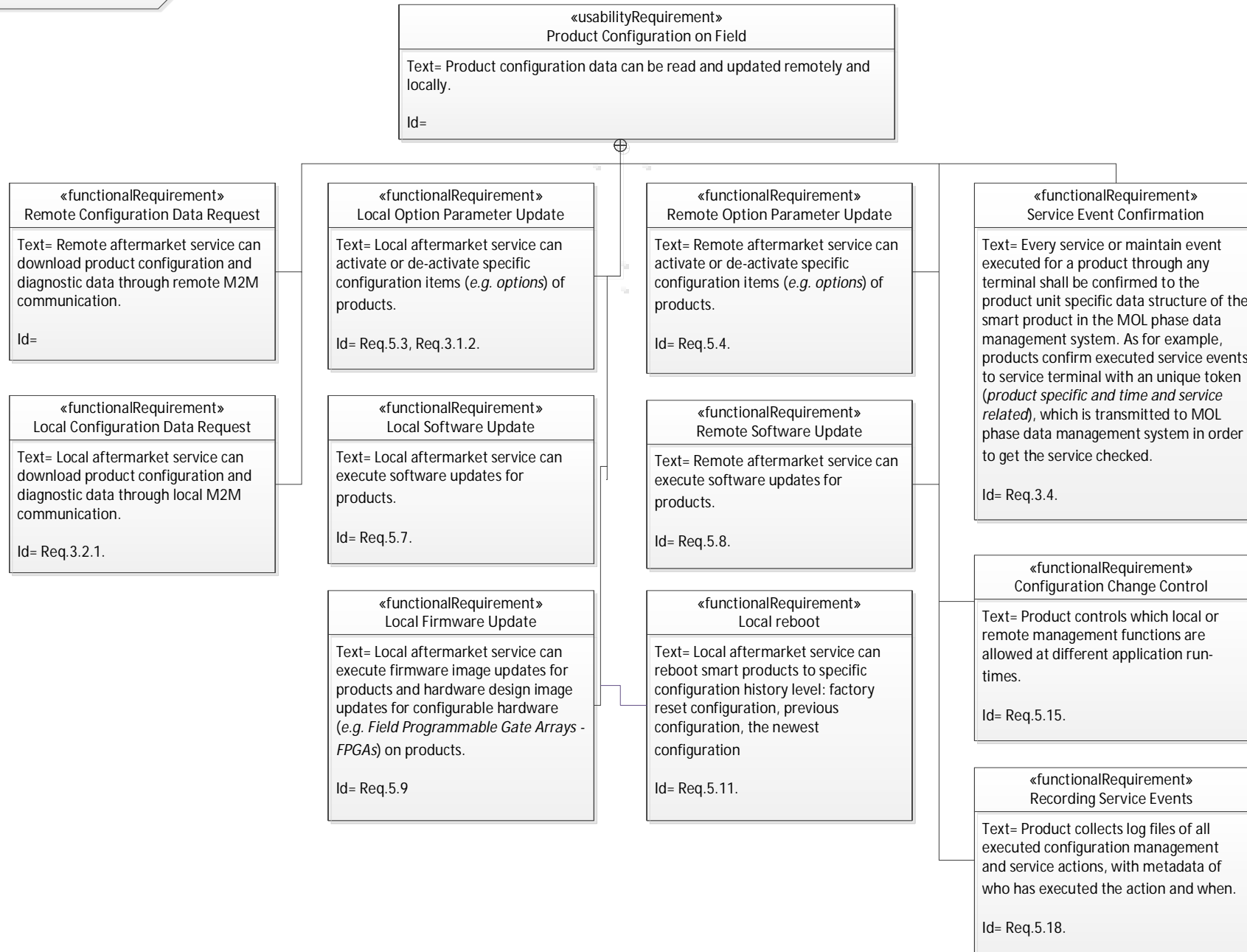


TITLE	ProMoNet DCM Requirements		DESCRIPTION	Requirement diagrams of ProMoNet Dynamic Configuration Management system for reconfigurable networked industrial products		
FILENAME	PROMONET_DCM_SYSTEM_REQUIREMENTS_12.VSD		DRAWN BY	PARKKILA TOMMI	DATE	4/12/2012
			PAGE	1 OF 6	REVISED	4/16/2013

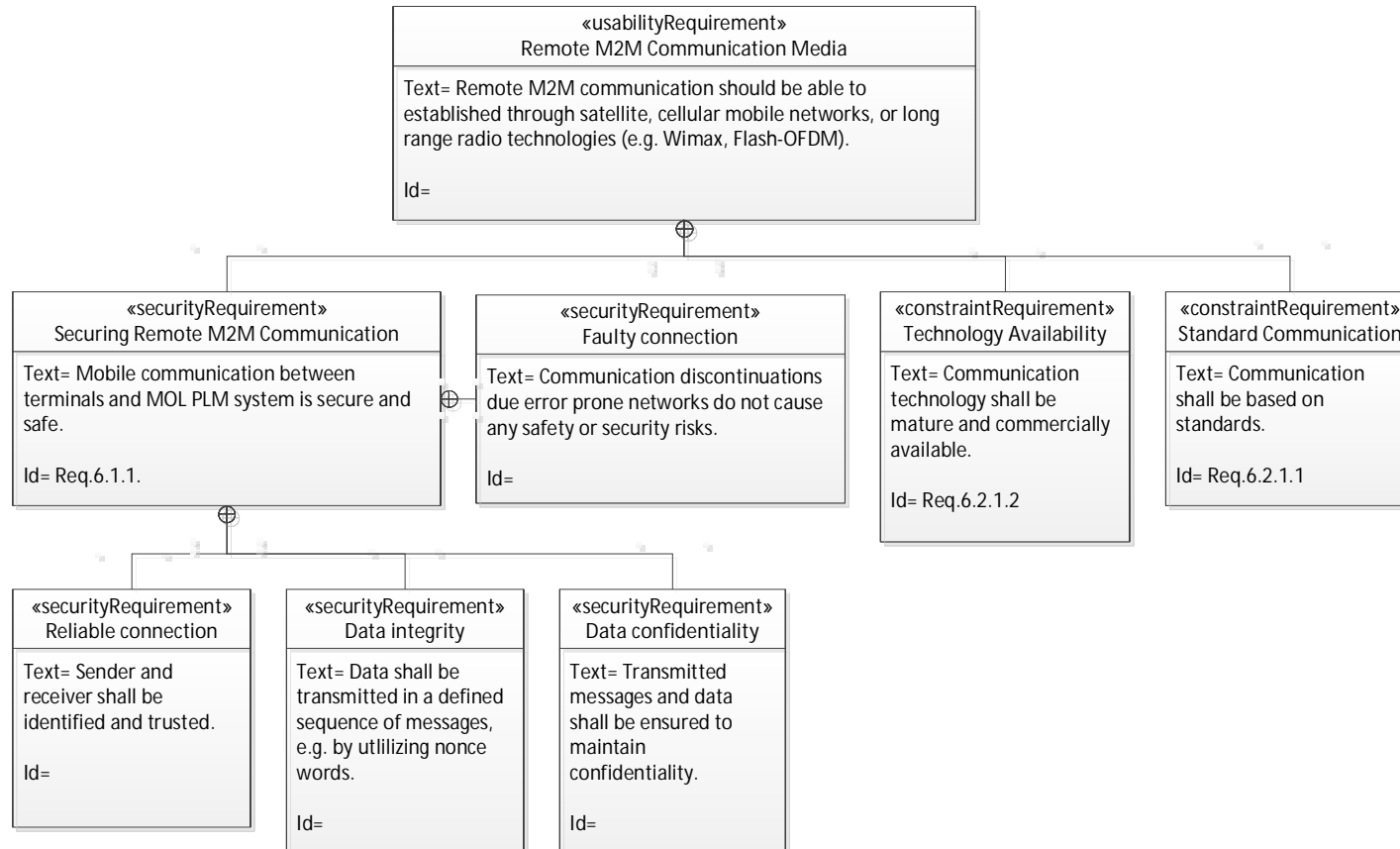
REVISIONS			
REV.	DESCRIPTION	DATE	BY
1.0	First version of the requirements created	4/25/2012	TOP
1.1			TOP
1.2	Added references to key requirements	4/8/2013	PEI



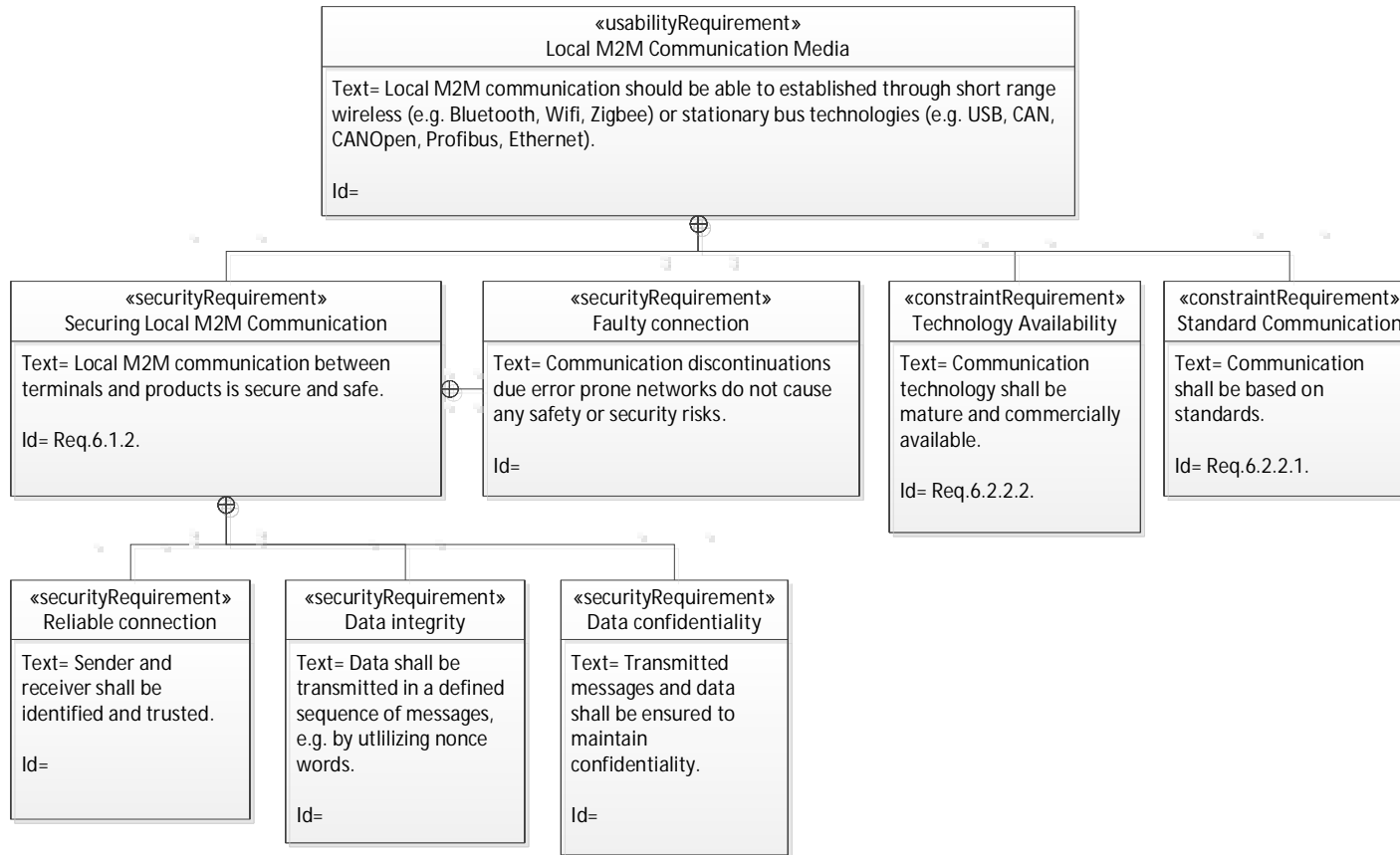
FILENAME PROMONET_DCM_SYSTEM_REQUIREMENTS_12.VSD	DRAWN BY PARKKILA TOMMI	DATE 4/12/2012
	PAGE 2 OF 6	REVISED 4/16/2013



FILENAME	DRAWN BY	DATE
PROMONET_DCM_SYSTEM_REQUIREMENTS_12.VSD	PARKKILA TOMMI	4/12/2012
	PAGE	REVISED
	3 OF 6	4/16/2013



FILENAME	DRAWN BY	DATE
PROMONET_DCM_SYSTEM_REQUIREMENTS_12.VSD	PARKKILA TOMMI	4/12/2012
	PAGE	REVISED
	4 OF 6	4/16/2013



FILENAME PROMONET_DCM_SYSTEM_REQUIREMENTS_12.VSD	DRAWN BY PARKKILA TOMMI	DATE 4/12/2012
	PAGE 5 OF 6	REVISED 4/16/2013

«usabilityRequirement»  
Evolving Product Unit Specific Data Structure

Text= Products have evolving as-maintained product unit specific data structures in MOL PLM system.

Id= Req.1.1.

«usabilityRequirement»  
Data structure replication

Text= Product unit specific data structure is replicated among the product and enterprise information system used (e.g. PLM, PDM, ERP)

Id= Req.1.6.

«functionalRequirement»  
Data generation

Text= OEM Production can generate the product unit specific data structure from the product's beginning of life (BOL) phase data management systems using customer order specifications.

Id= Req.1.2.

«functionalRequirement»  
Update Product Data Structure

Text= Product specific data structure in MOL phase data management system can be updated.

Id=

«usabilityRequirement»  
Multi-organizational access

Text= Third party organizations can have restricted access to review the product unit specific data structure.

Id= Req.2.2.

«functionalRequirement»  
Data Download

Text= Local aftermarket service can download product unit specific configuration data to a mobile service terminal through remote M2M communication.

Id= Req.3.1.1.

«functionalRequirement»  
Aftermarket update

Text= Remote aftermarket service can update product specific data structure.

Id= Req.1.8.

«functionalRequirement»  
Local service update

Text= Local aftermarket service can upload and update product diagnostic data and configuration data to product unit specific data structure through remote M2M communication.

Id= Req.3.2.2.

«functionalRequirement»  
Maintaining configuration history

Text= Product data structure stores at least three levels of configuration data history of configuration items as back-up-data.

Id= Req.1.11.

«usabilityRequirement»  
Configuration Data Granulativity

Text= Product configuration data shall be described in details, which includes five levels of configuration items; 1.physical hardware, 2. hardware designs, 3. firmware software, 4. application software, and 5. parameters.

Id=

«functionalRequirement»  
BOL update

Text= Remote aftermarket service can bring and add new data from BOL phase data management system.

Id= Req.1.7.

«functionalRequirement»  
Data update

Text= Remote aftermarket service can update data values or add new data to product unit specific data structure.

Id=

«functionalRequirement»  
Log Configuration Change Events

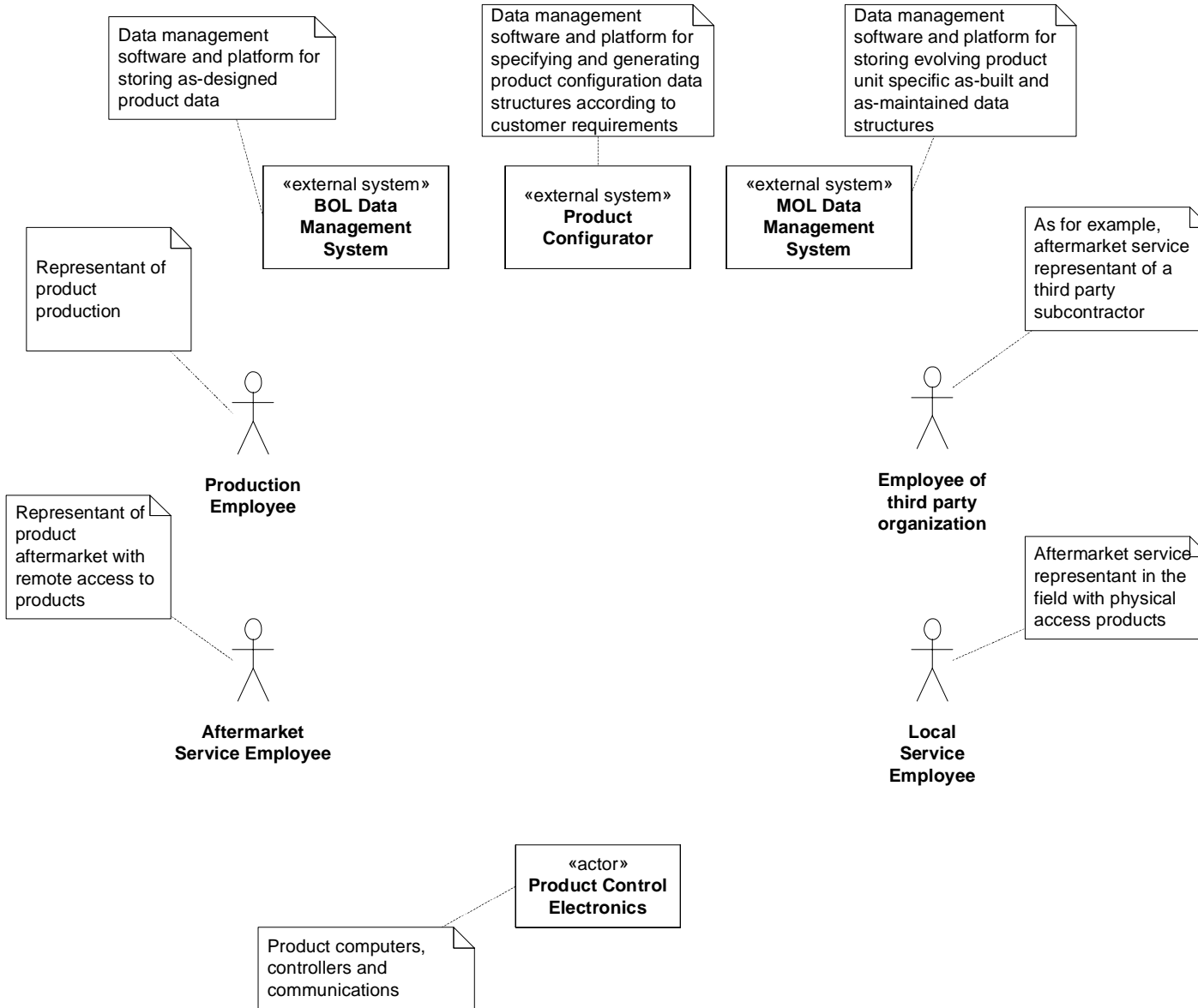
Text= MOL PLM System keeps a service log file of every configuration change event, with a meta information of the date and location of the change, and information of the party who executed the change.

Id= Req.1.10.

FILENAME PROMONET_DCM_SYSTEM_REQUIREMENTS_12.VSD	DRAWN BY PARKKILA TOMMI	DATE 4/12/2012
	PAGE 6 OF 6	REVISED 4/16/2013

TITLE	ProMoNet DCM System Use Cases		DESCRIPTION	Use case diagrams of ProMoNet dynamic configuration management (DCM) system for reconfigurable networked industrial products	
FILENAME	PROMONET_DCM_USECASES_12.VSD		DRAWN BY	PARKKILA TOMMI, ISTO PEKKA	
			DATE	5/8/2012	
			PAGE	1 OF 26	
			REVISED	3/12/2013	

REVISIONS			
REV.	DESCRIPTION	DATE	BY
0.1	First draft created	05/15/2012	TOP
0.2	Steps -- > Essence, production use cases mergerd, identify employee cases added	05/16/2012	TOP
0.3	Primary Use Cases, systemProcess act-diagrams, added, some name changes	05/17/2012	TOP
0.4	Local Service Use Cases Edited	08/23/2012	TOP
0.5	Aftermarket Service Use cases edited	09/10/2012	TOP
0.6	Local Service Use Cases Synchronized with Flows (rev 0.4)	09/26/2012	TOP
0.7	Renamed use case and information flows. Minor modifications to use cases.	11/22/2012	PEI
0.8	Aligned to domain knowledge rev. 0.6.	11/23/2012	PEI
0.9	Added LSE Retrieve/Store Service Code. Aligned to flows 0.8. Improved ASE cases.	02/06/2013	PEI
10	Added DCM Usage, improved LSE cases, redesigned and renamed ASE cases.	02/24/2013	PEI
11	Improved all cases.	02/28/2013	PEI
12	Corrected for review findings: Local reboot, versioning and Product Configurator.	03/12/2013	PEI



FILENAME	PROMONET_DCM_USECASES_12.VSD	DRAWN BY	DATE
		PARKKILA TOMMI, ISTO PEKKA	5/8/2012
		PAGE	REVISED
		2 OF 26	3/12/2013



uc Primary Use Cases [DCM System]

Representant of product production

Production Employee

Create Product Specific Data Structure

Create Product Specific Configuration

As for example, aftermarket service representant of a third party subcontractor

Employee of third party organization

Retrieve Current Configuration From PLM

Representant of product aftermarket. Remotely configures networked products.

Aftermarket Service Employee

Make New Configuration Change Task

Read Current Configuration From PLM

Pull Current Configuration From Product

Check Update Status

Push Product Configuration Update

Write Configuration Data To PLM

Aftermarket service representant in the field next to products. Configures non-networked products.

Local Service Employee

Retrieve Service Code

Store Service Code

Get Configuration Update Receipt

Retrieve Current Configuration Data From PLM

Get Current Configuration Data From Product

Retrieve Configuration Update Package

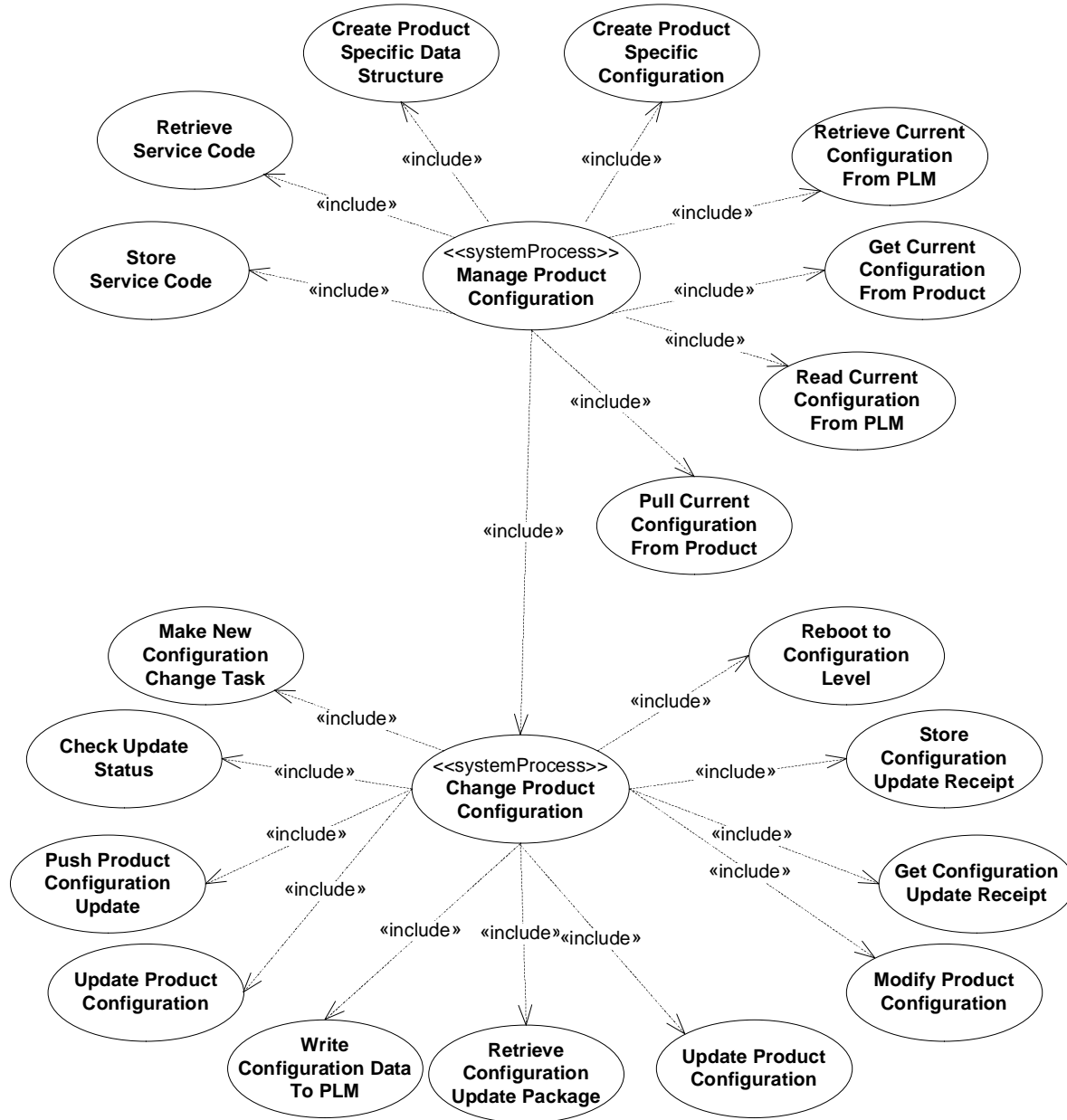
Update Product Configuration

Modify Product Configuration

Store Configuration Update Receipt

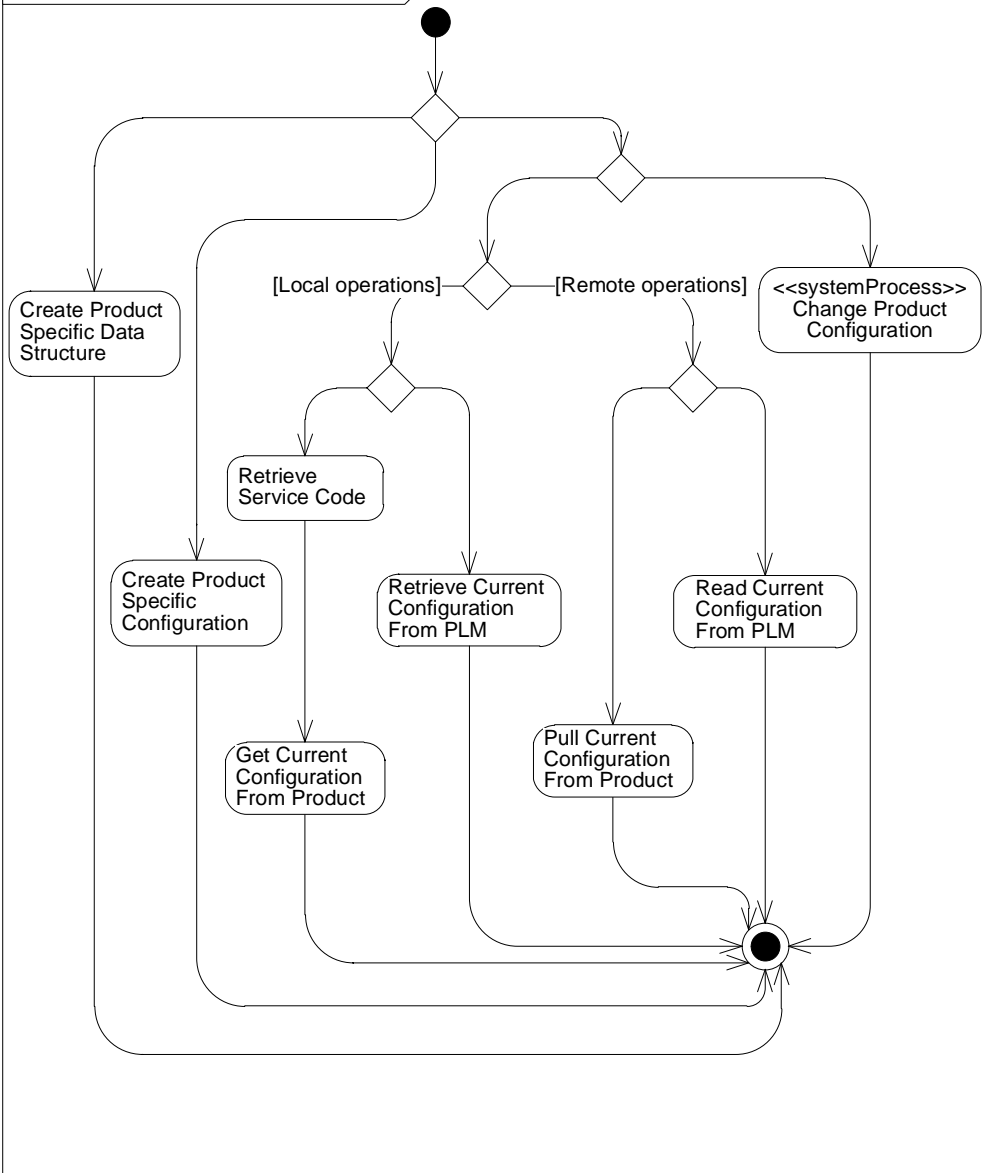
Reboot to Configuration Level

FILENAME PROMONET_DCM_USECASES_12.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 5/8/2012
	PAGE 3 OF 26	REVISED 3/12/2013

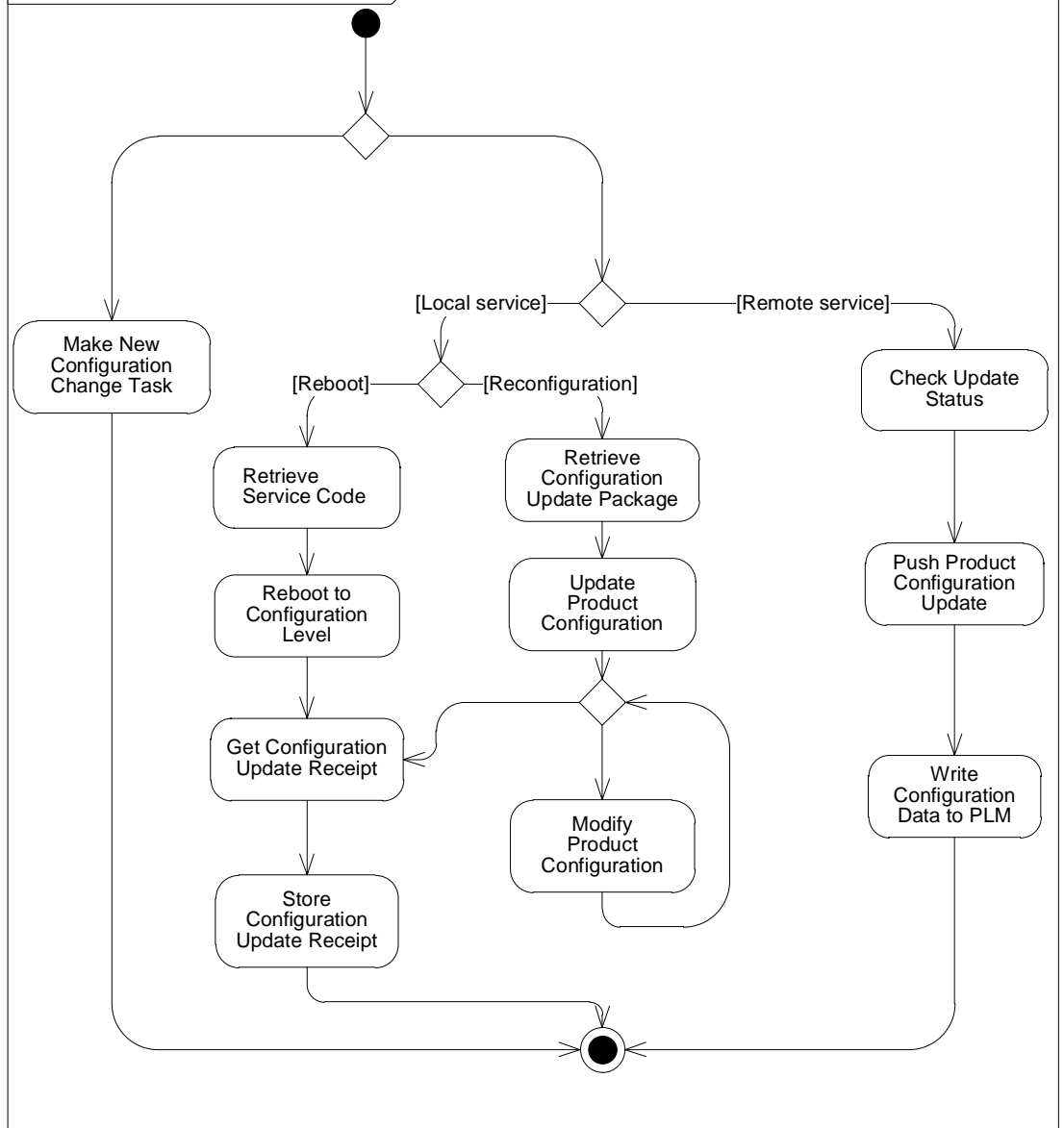


FILENAME	DRAWN BY	DATE
PROMONET_DCM_USECASES_12.VSD	PARKKILA TOMMI, ISTO PEKKA	5/8/2012
	PAGE	REVISED
	4 OF 26	3/12/2013

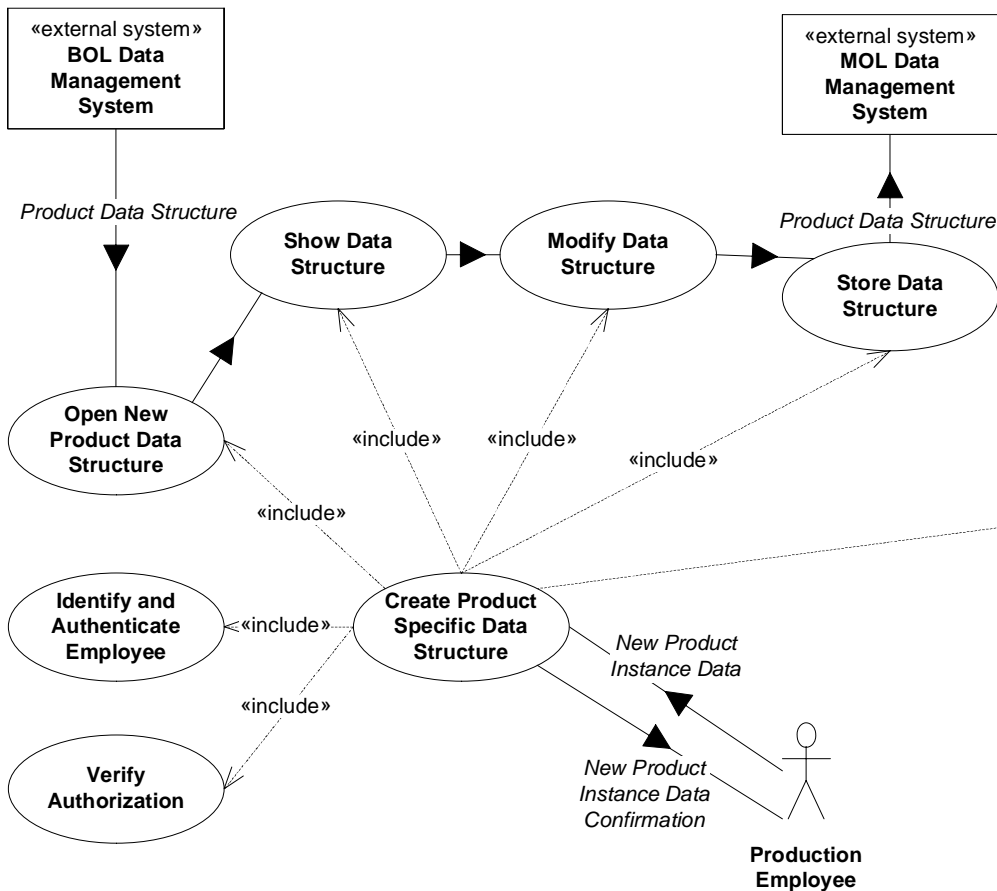
<<systemProcess>>  
act Manage Product Configuration



<<systemProcess>>  
act Change Product Configuration



FILENAME	DRAWN BY	DATE
PROMONET_DCM_USECASES_12.VSD	PARKKILA TOMMI, ISTO PEKKA	5/8/2012
	PAGE	REVISED
	5 OF 26	3/12/2013



### Create Product Specific Data Structure

Narrative Description:

Production employee wants to create a new product specific data structure. Production employee sends "New Product Instance Data" request to DCM system. DCM system identifies and authenticates the employee (by e.g. user name and password) and verifies that the employee has authorization to execute the request and if so opens new product data structure, and fills it with default configuration values. DCM system then shows product data structure for production employee to modify and stores the created product specific data structure to MOL Data Management System. DCM system indicates the completion of the request with "New Product Instance Data Confirmation" message.

Description:

Production employee creates new product specific data structure.

Actors:

Production Employee

BOL Data Management System

MOL Data Management System

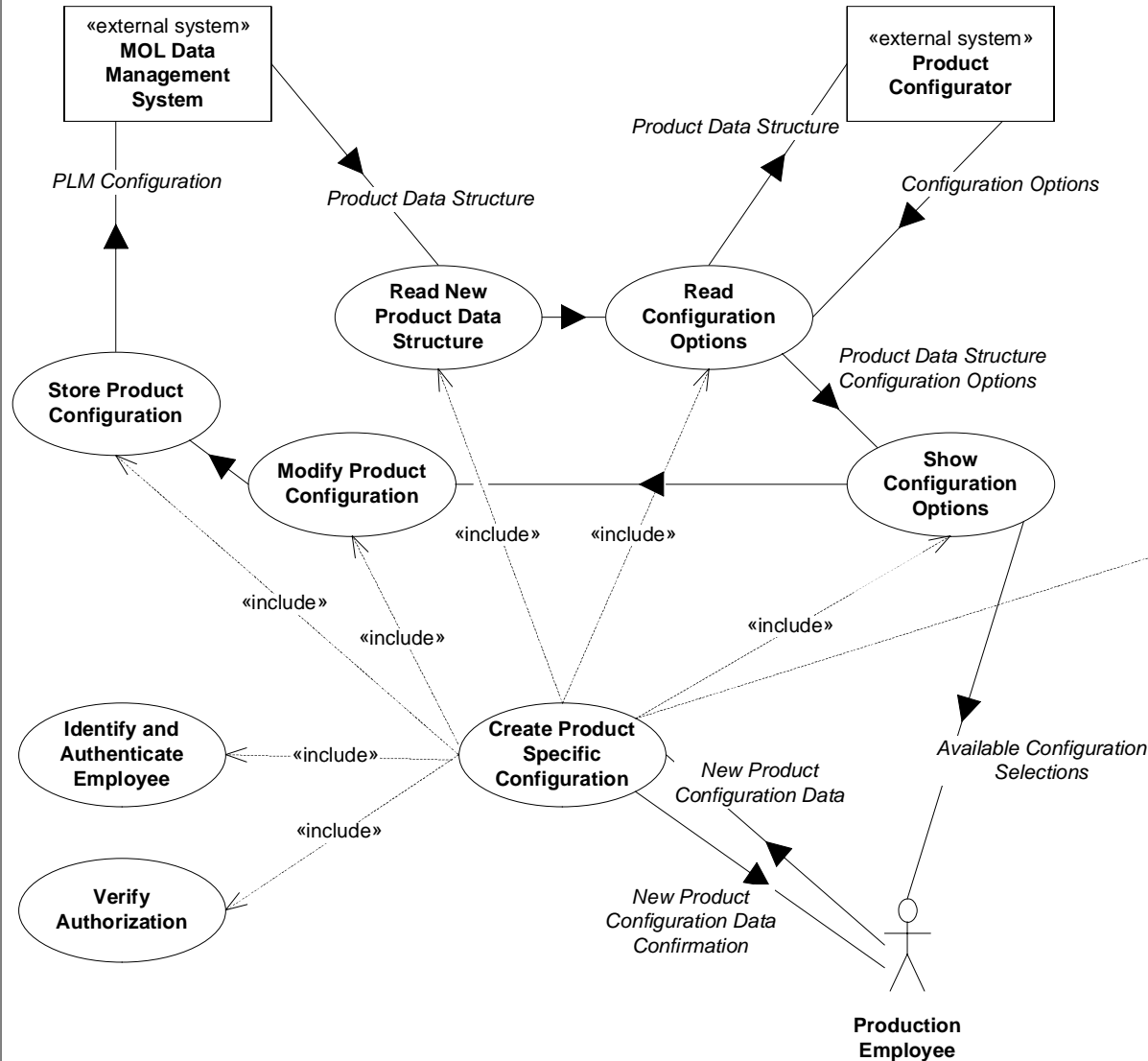
Preconditions/Assumptions:

DCM system has an access control for production service employee from the internal network.

Essence

1. Send New Product Instance Data
2. Identify and Authenticate Employee
3. Verify Authorization for the Request
4. Open New Product Data Structure
5. Show Data Structure
6. Modify Data Structure
7. Store Data Structure
8. Return New Product Instance Data Confirmation

FILENAME	PROMONET_DCM_USECASES_12.VSD	DRAWN BY	PARKKILA TOMMI, ISTO PEKKA	DATE	5/8/2012
		PAGE	6 OF 26	REVISED	3/12/2013



## Create Product Specific Configuration

Narrative Description:

Production employee wants to create a new product specific configuration data to PLM. Production employee sends "New Product Configuration Data" request to DCM system. DCM system identifies and authenticates the employee (by e.g. user name and password) and verifies that the employee has authorization to execute the request and if so opens new product data structure and fills it with default configuration values. DCM system then shows product data structure and available configuration options for it to the production employee for modification and stores the created product specific configuration data to MOL Data Management System. DCM system indicates completion with "New Product Configuration Data Confirmation" message.

Description:

Production employee creates new product specific configuration.

Actors:

Production Employee  
MOL Data Management System  
Product Configurator

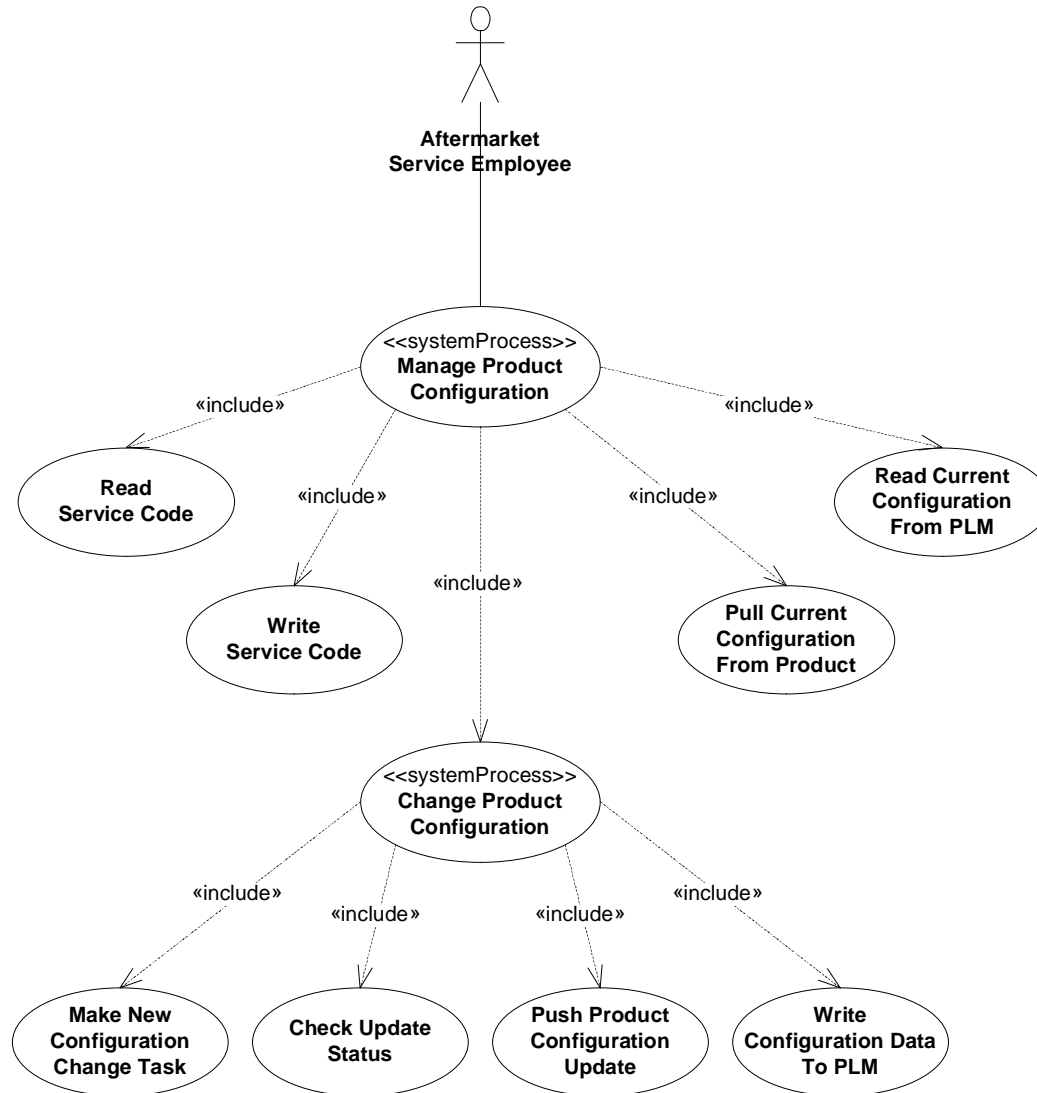
Preconditions/Assumptions:

DCM system has an access control for production service employee from the internal network.

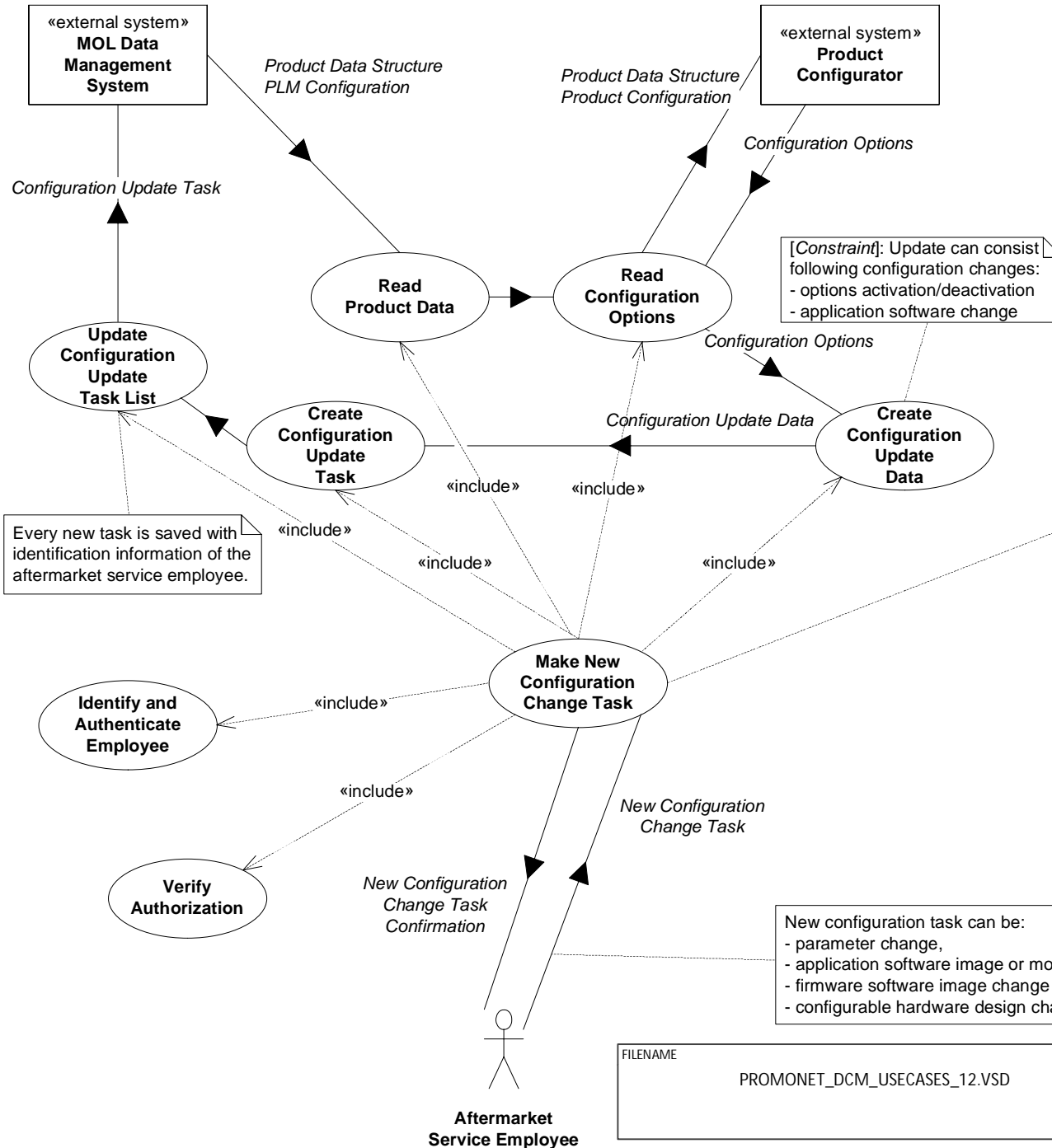
Essence

1. Send New Product Configuration Data
2. Identify and Authenticate Employee
3. Verify Authorization for the Request
4. Read Product Data Structure
5. Read Configuration Options
6. Show Product Data Structure with Configuration Options
7. Modify product configuration
8. Store product configuration to MOL as PLM Configuration
9. Return New Product Configuration Data Confirmation

FILENAME	DRAWN BY	DATE
PROMONET_DCM_USECASES_12.VSD	PARKKILA TOMMI, ISTO PEKKA	5/8/2012
	PAGE	REVISED
	7 OF 26	3/12/2013



FILENAME	PROMONET_DCM_USECASES_12.VSD	DRAWN BY	PARKKILA TOMMI, ISTO PEKKA	DATE	5/8/2012
		PAGE	8 OF 26	REVISED	3/12/2013



## Make New Configuration Change Task

### Narrative Description:

Aftermarket service employee wants to propose new configuration change task for a specific product. Aftermarket service employee sends "New Configuration Change Task" request to DCM system. DCM system identifies (e.g. by user name) and authenticates the employee (e.g. by password) and verifies that the employee has authorization to submit new configuration update task. If authorized, DCM system allows aftermarket service employee to create Configuration Update Task including the changed artifacts, and new parameter values, and identification information of the employee. DCM system inserts the new task to Configuration Update Task List of the corresponding product in MOL Data Management System. DCM system returns "New Configuration Change Task Confirmation" after successful task list update.

### Description:

Aftermarket employee adds a new configuration update task to product specific configuration update task list.

### Actors:

Aftermarket employee  
MOL Data Management System  
Product Configurator

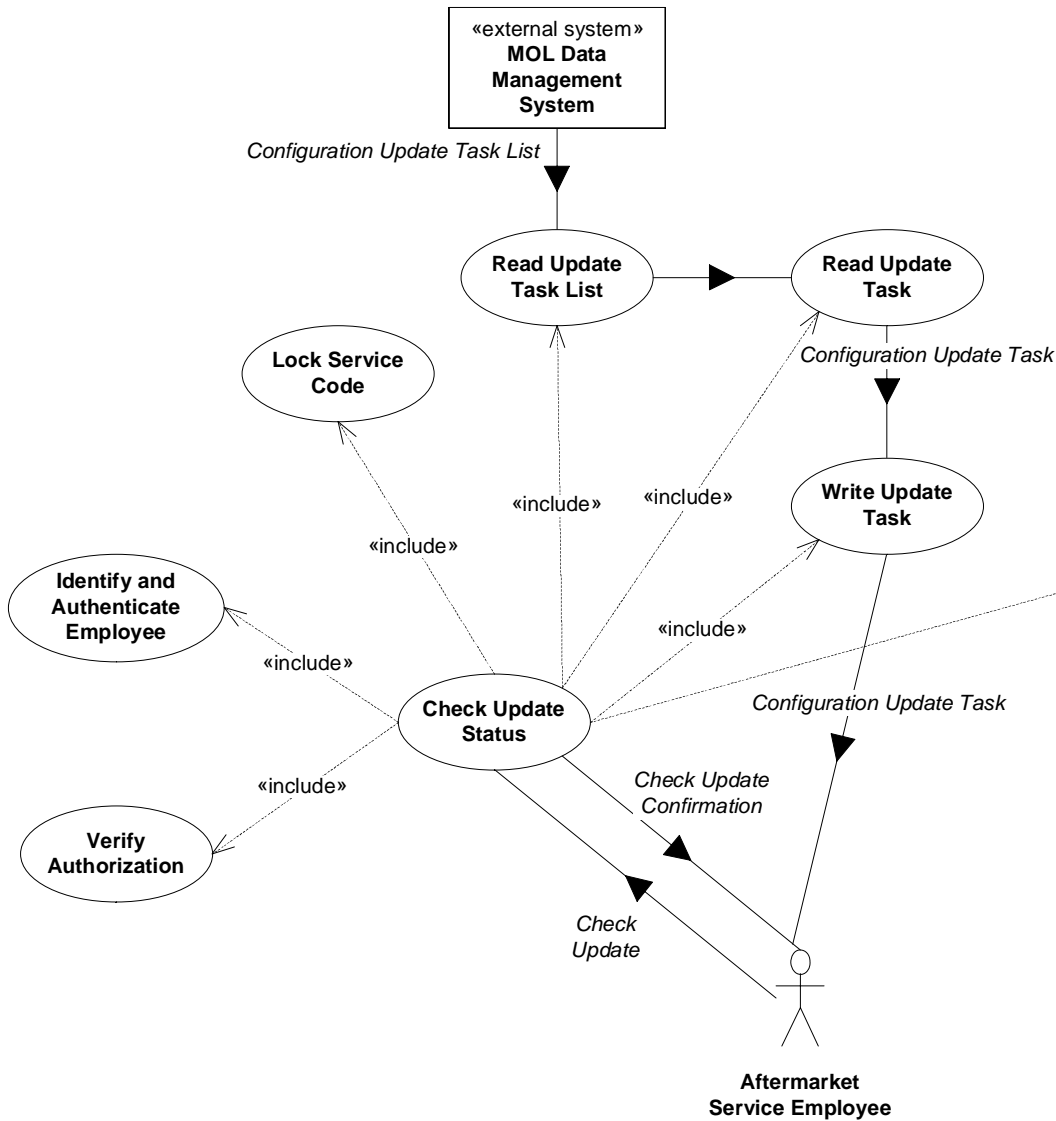
### Preconditions/Assumptions:

DCM system has an access control for aftermarket service employee from the internal network.

### Essence:

1. Send New Configuration Change Task
2. Identify and authenticate employee
3. Verify Authorization for creating a task
4. Read product data structure and configuration from MOL DMS
5. Read Configuration Options from Product Configurator
6. Create Configuration Update Data
7. Create new Configuration Update Task
8. Insert the new update task to Configuration Update Task List
9. Send New Configuration Change Task Confirmation

FILENAME	DRAWN BY	DATE
PROMONET_DCM_USECASES_12.VSD	PARKKILA TOMMI, ISTO PEKKA	5/8/2012
	PAGE	REVISED
	9 OF 26	3/12/2013



## Check Update Status

### Narrative Description:

Aftermarket service employee wants to check if any configuration updates are pending for a product in Configuration Update Task List and if so prepare for executing an update. Aftermarket service sends "Check Update" request to the DCM system. DCM System identifies and authenticates the employee, and if the employee is authorized for the Check Update request and an update is pending, retrieves the current Service Code, locks it and returns one of the Configuration Update Tasks in the Configuration Update Task List. DCM System stores Configuration Update Task including the service code to Configuration Storage and returns "Check Update Confirmation".

### Description:

Aftermarket service requests a configuration update task from the list of pending tasks and stores it in Configuration Storage.

### Actors:

Aftermarket Service Employee  
MOL Data Management System

### Preconditions/Assumptions:

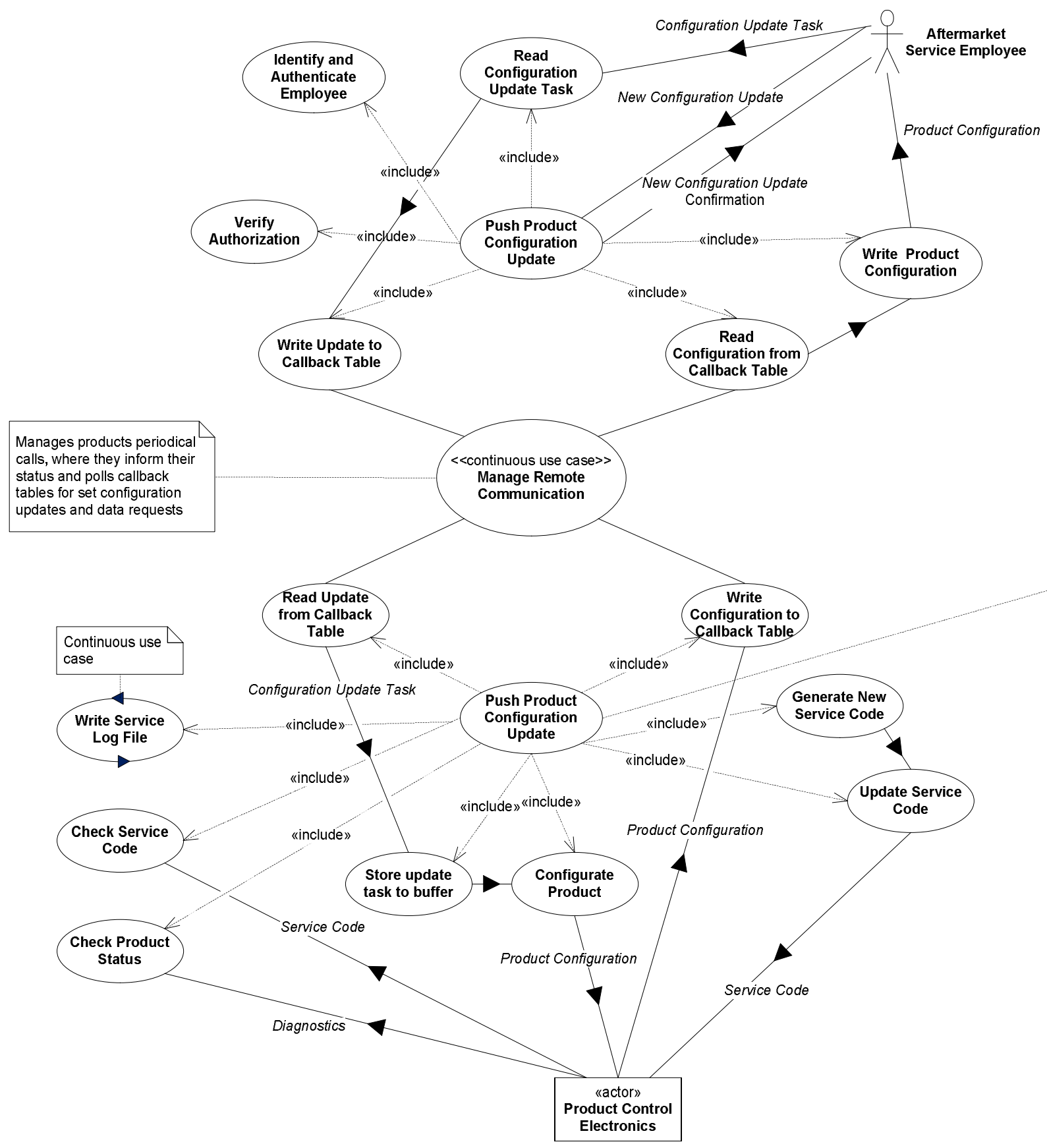
DCM system has an access control for aftermarket service employee from the internal network. Aftermarket service employee has a device or service that provides Configuration Storage for the configuration related data. Service Code pair consist of unique product ID number and random service code (PIN), which is updated by the product after every successful configuration update.

### Essence:

1. Send Check Update
2. Identify and authenticate employee
3. Verify authorization for Check Update request
4. Lock service code
5. Read Configuration Update Task List from MOL DMS
6. Read update task from configuration update list
7. Write update task to configuration storage
8. Return Check Update Confirmation

FILENAME PROMONET_DCM_USECASES_12.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 5/8/2012
	PAGE 10 OF 26	REVISED 3/12/2013





### Push Product Configuration Update

**Narrative Description:**  
 Aftermarket service employee wants to update product configuration with configuration update task stored in Configuration Storage. The employee sends "New Configuration Update" request to DCM system. DCM system identifies and authenticates the employee and, if the employee has authorization for the request, reads the Configuration Update Task with current Service Code from Configuration Storage and inserts it into product Callback Table. Product polls its callback table periodically, when it detects new Configuration Update Task, product performs the configuration update. The product checks first the validity of the received service code by comparing it to the current service code stored into local memory of the product. If the secure codes match, product stores the Configuration Update Data from Configuration Update Task in local buffer memory, checks its status for executing the configuration update, and updates configuration with new configuration settings when possible. After succesful configuration update, product generates new service codes and returns Product Configuration message including service codes and new current configuration to Configuration Storage. DCM system clears the task from Callback Table and returns "New Configuration Update Confirmation".

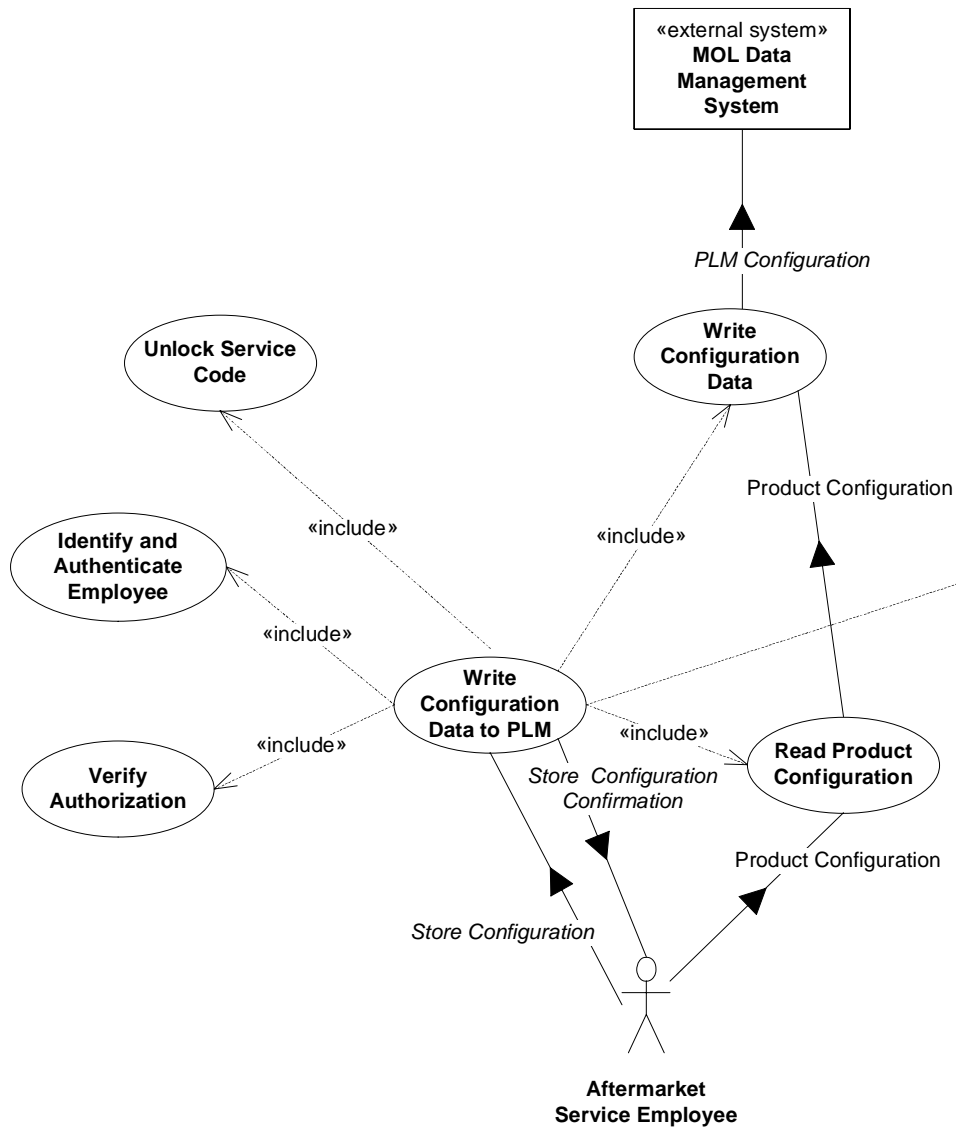
**Description:**  
 Aftermarket service employee updates configuration of a product on field.

**Actors:**  
 Aftermarket service employee  
 Product Control Electronics

**Preconditions/Assymptions:**  
 Aftermarket service employee has a device or service that provides Configuration Storage for the configuration related data. DCM system has an access control for aftermarket service employee from the internal network and manages secure mobile connections to products. Products have mobile remote connections available. Product has local memory to store the configuration update data until update is possible and safe.

- Essence:**
1. Send New Configuration Update
  2. Identify and authenticate employee
  3. Verify authorization for New Configuration Update
  4. Read configuration update task from configuration storage
  5. Write update task to Callback Table
  6. Read update task from Callback Table
  7. Check service code
  8. Store configuration update task to local buffer memory
  9. Check product status
  10. Configure product
  11. Generate new service code
  12. Update service code in product configuration
  13. Write Product Configuration to Callback Table
  14. Read Product Configuration from Callback Table
  15. Write Product Configuration to configuration storage
  16. Send New Configuration Update Confirmation

FILENAME	PROMONET_DCM_USECASES_12.VSD	DRAWN BY	PARKKILA TOMMI, ISTO PEKKA	DATE	5/8/2012
		PAGE	11 OF 26	REVISED	3/12/2013



## Write Configuration Data to PLM

### Narrative Description:

Aftermarket service employee wants to upload and update current configuration data read from a product to MOL Data Management System. The aftermarket service employee sends "Store Configuration" message to the DCM system. DCM system Identifies and authenticates the employee and if the employee has authorization for the request, DCM system reads the current configuration of the product including new service code stored in Configuration Storage and stores them to MOL Data Management System, unlocks the service code, and returns "Store Configuration Confirmation" message to the aftermarket service employee.

### Description:

Aftermarket service updates product configuration data in Configuration Storage into MOL Data Management System.

### Actors:

Aftermarket Service Employee  
MOL Data Management System

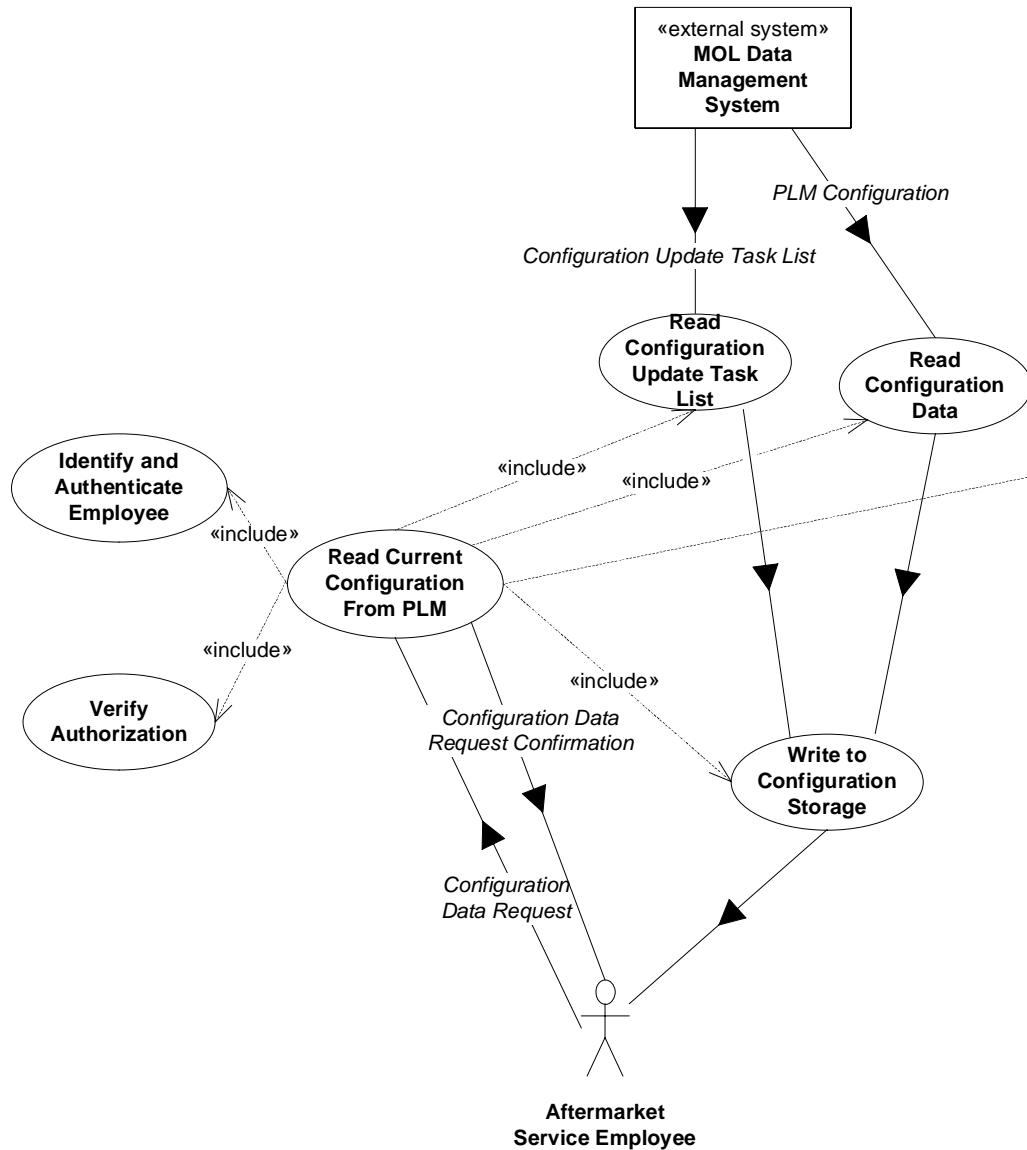
### Preconditions/Assumptions:

DCM system has an access control for aftermarket service employee from the internal network. Aftermarket service employee has a device or service that provides Configuration Storage for the configuration related data. Service code pair consist of unique product ID number and random service code (PIN), which is updated by the product after every successful configuration update.

### Essence:

1. Send Store Configuration
2. Identify and authenticate employee
3. Verify authorization for the request
4. Read Product Configuration from Configuration Storage
5. Write Product Configuration including Service Code to MOL DMS
6. Unlock service code
7. Send Store Configuration Confirmation

FILENAME PROMONET_DCM_USECASES_12.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 5/8/2012
	PAGE 12 OF 26	REVISED 3/12/2013



### Read Current Configuration From PLM

*Narrative Description:*

Aftermarket service employee wants to download current configuration data of a product from MOL Data Management System. Aftermarket service employee sends "Configuration Data Request" message to the DCM system. DCM system identifies and authenticates the employee, and if the employee has authorization for the request, reads corresponding product configuration and configuration update task list from MOL Data Management System and writes them to configuration storage. DCM system indicates completion with a "Configuration Data Request Confirmation" message.

*Description:*

Aftermarket service employee reads current configuration and update tasks list from PLM to Configuration Storage.

*Actors:*

Aftermarket service employee  
MOL Data Management System

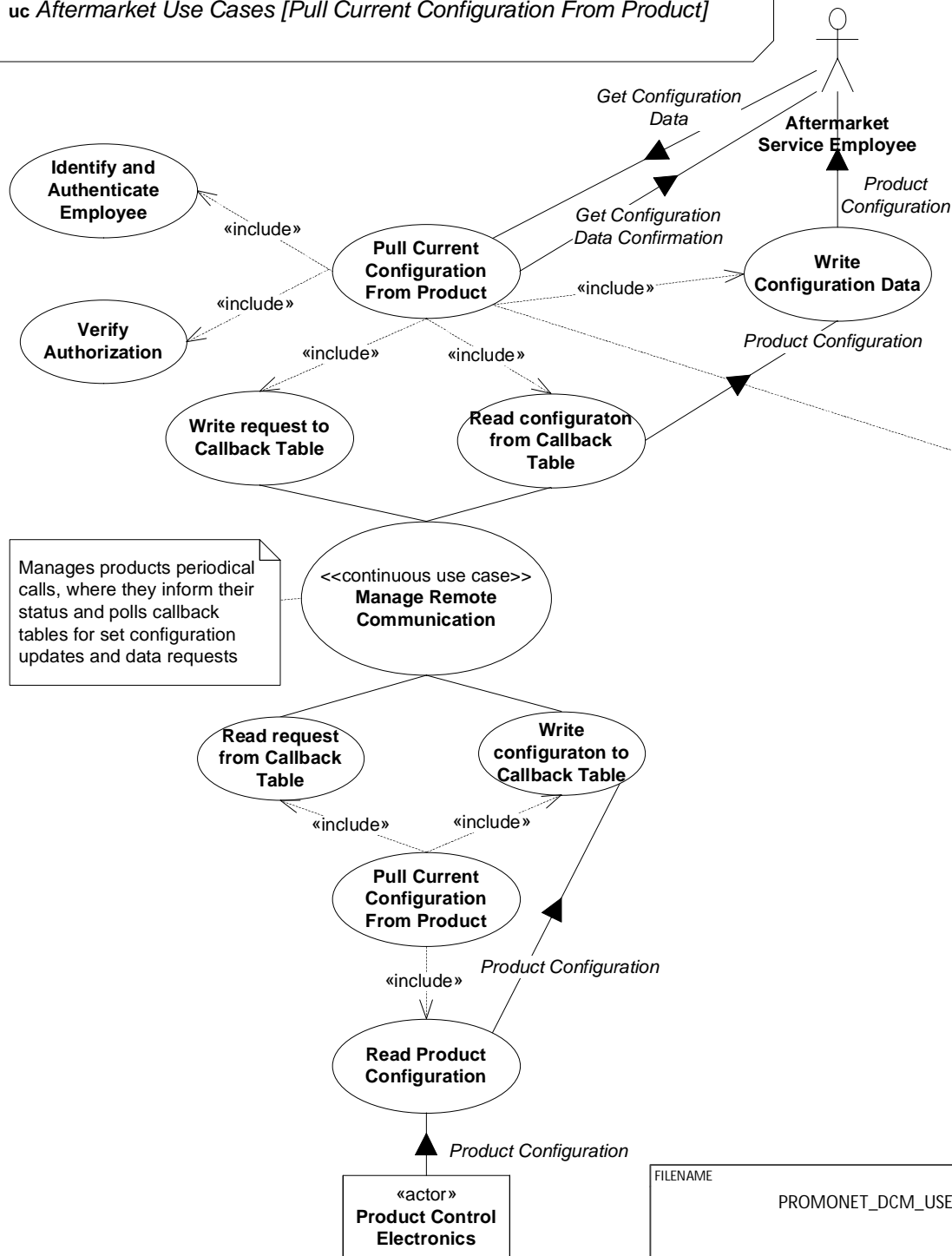
*Preconditions/Assumptions:*

DCM system has an access control for aftermarket service employee from the internal network. Aftermarket service employee has a device or service that provides Configuration Storage for the configuration related data.

*Essence:*

1. Send Configuration Data Request
2. Identify and authenticate the employee
3. Verify authorization for the request
4. Read configuration data from MOL Data Management System
5. Read Configuration Update Task List
6. Write configuration data and task list to Configuration Storage
7. Send Configuration Data Request Confirmation

FILENAME	DRAWN BY	DATE
PROMONET_DCM_USECASES_12.VSD	PARKKILA TOMMI, ISTO PEKKA	5/8/2012
	PAGE	REVISED
	13 OF 26	3/12/2013



### Pull Current Configuration From Product

*Narrative Description:*

Aftermarket service employee wants to get current configuration data from a product on field and store it in Configuration Storage. Aftermarket service sends "Get Configuration Data" request to the DCM system. DCM system identifies and authenticates the employee, and if the employee has authorization for the request, updates callback table of the corresponding product with the data request information. Product polls its callback table periodically, when it detects the new data requests, product reads its current configuration data from its memory and sends the data to DCM system which writes it to Configuration Storage. DCM system clears the task from callback table and indicates completion with "Get Configuration Data Confirmation".

*Description:*

Aftermarket service employee requests current configuration data of a specific product from field through remote communication connection and writes it to Configuration Storage.

*Actors:*

Aftermarket service employee  
Product Control Electronics

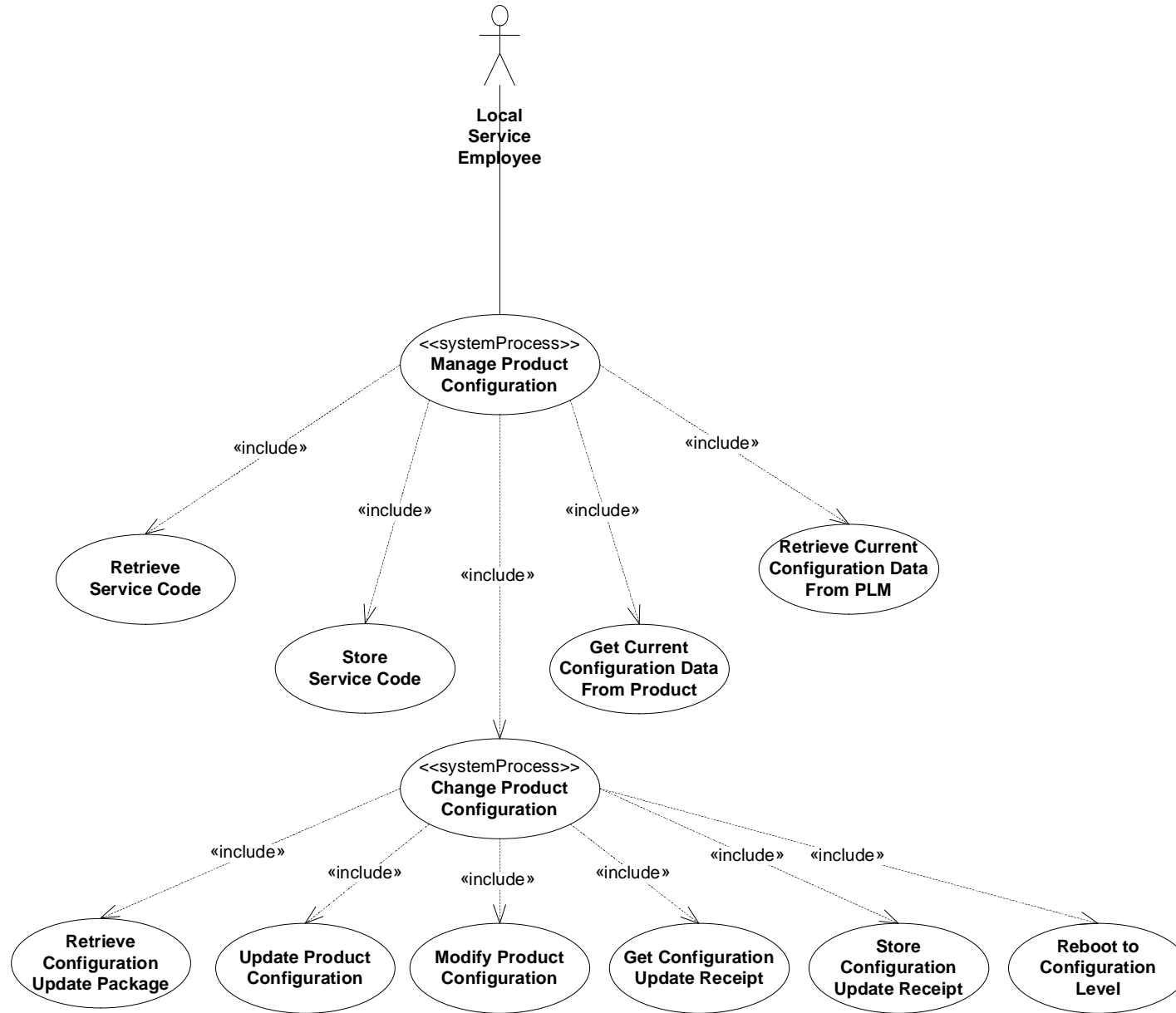
*Preconditions/Assumptions:*

DCM system has an access control for aftermarket service from the internal network. DCM system manages secure mobile connections (e.g. SSL) to products. Products have mobile remote connections available. Aftermarket service employee has a device or service that provides Configuration Storage for the configuration related data.

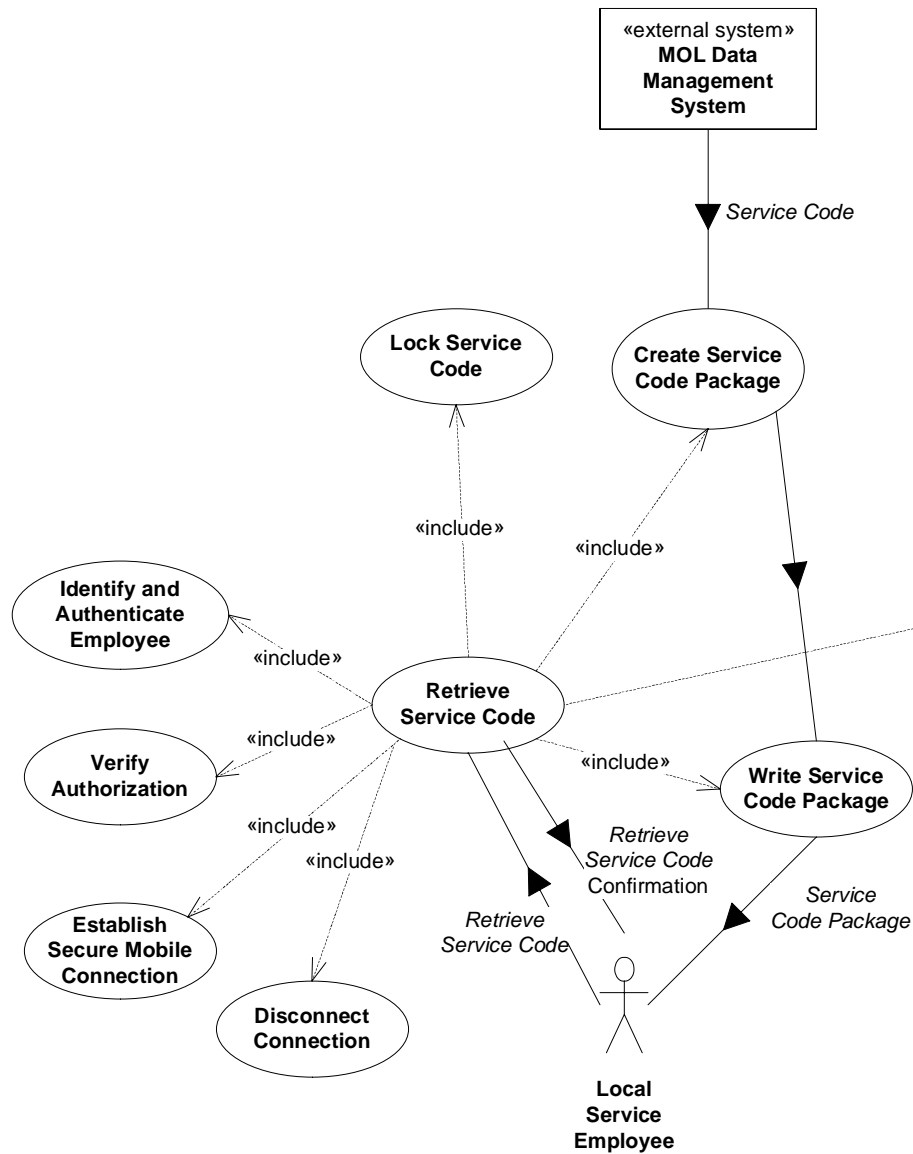
*Essence:*

1. Send Get Configuration Data
2. Identify and authenticate employee
3. Verify authorization for the request
4. Write Get Configuration Data request to Callback Table
5. Read Get Configuration Data request from Callback Table
6. Read product configuration
7. Write product configuration to Callback Table
8. Read product configuration from Callback Table
9. Write product configuration to configuration storage
10. Send Get Configuration Data Confirmation

FILENAME PROMONET_DCM_USECASES_12.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 5/8/2012
	PAGE 14 OF 26	REVISED 3/12/2013



FILENAME	PROMONET_DCM_USECASES_12.VSD	DRAWN BY	PARKKILA TOMMI, ISTO PEKKA	DATE	5/8/2012
		PAGE	15 OF 26	REVISED	3/12/2013



## Retrieve Service Code

### Narrative Description:

Local service employee wants to download service codes for a product to be serviced. Product is in the field beyond straight mobile communication network coverage. Using service terminal the local service establish a secure mobile communication connection to the DCM system. Then the employee sends to the DCM system "Retrieve Service Code" message including the specific service request to be performed. DCM system authenticates the employee, checks which product and request the employee is authorized for. If the employee is authorized for the given product and request and the service code has not being locked earlier by any other actor, DCM system writes service code and the request data to mobile storage using the service terminal, indicates completion to the employee with "Retrieve Service Code Confirmation" and closes the connection.

### Description:

Local service retrieves service code for a service request to be executed for a product on field.

### Actors:

Local Service Employee  
MOL Data Management System

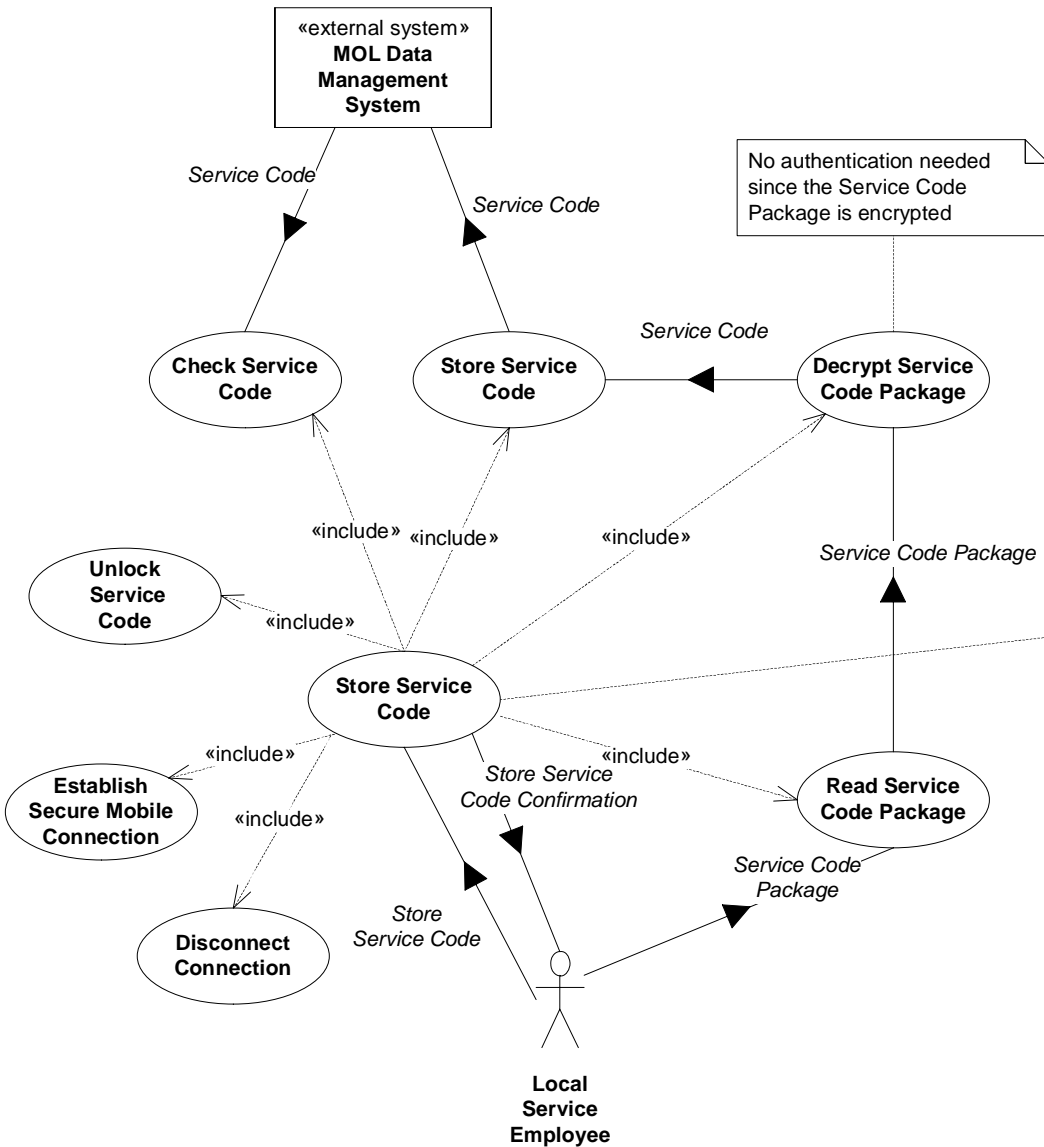
### Preconditions/Assumptions:

Local service has a service terminal, which is equipped with a mobile communication interface (e.g. cellular radio, satellite, or LTE), and which has memory and other resources enough to run a communication application. Service terminal also has memory to store service code package containing the service code and the request. That storage may be integrated to the service terminal if terminal is mobile, otherwise the storage is detachable and mobile such as USB memory stick. Service code data is encrypted.

### Essence:

1. Establish secure mobile connection
2. Send Retrieve Service Code
3. Identify and authenticate employee
4. Verify authorization
5. Lock service code
6. Create Service Code Package
7. Write Service Code Package to mobile storage
8. Send Retrieve Service Code Confirmation
9. Disconnect Connection

FILENAME PROMONET_DCM_USECASES_12.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 5/8/2012
	PAGE 16 OF 26	REVISED 3/12/2013



### Store Service Code

*Narrative Description:*

Local service employee wants to upload and update service code of a product to MOL Data Management System. The employee establishes a secure mobile communication connection to the DCM system. Then the employee sends "Store Service Code" message including employee identification to DCM system. The service terminal reads Service Code Package from the mobile storage and sends it to DCM system. DCM system receives and decrypts it and checks that the old service code stored in the Service Code Package and service code stored in the MOL Data Management system match. DCM system stores new service code to MOL Data Management system, unlocks the service code, and returns "Service Code Confirmation" message to local service and closes the connection.

*Description:*

Local service updates product service code stored into MOL Data Management System.

*Actors:*

- Local Service Employee
- MOL Data Management System

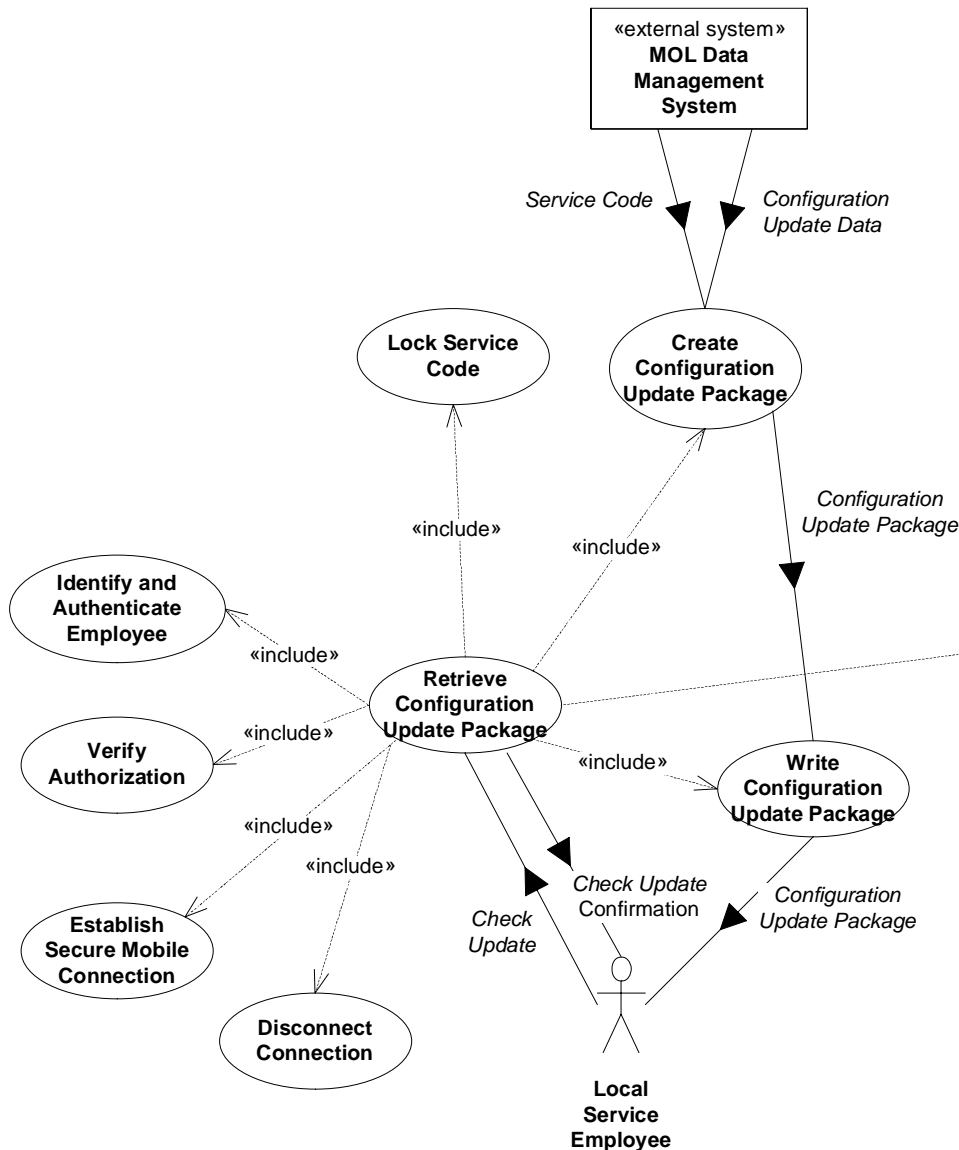
*Preconditions/Assumptions:*

Local service has a service terminal, which is equipped with a mobile communication interface (e.g. cellular radio, satellite, or LTE), and which has memory and other resources enough to run a communication application. Service terminal also has memory to store service code package. That storage may be integrated to the service terminal if terminal is mobile, otherwise the storage is detachable and mobile such as USB memory stick.

*Essence:*

1. Establish secure mobile connection
2. Send Store Service Code
3. Read Service Code Package from mobile storage
4. Decrypt service Code Package
5. Check service code
6. Store updated service code
7. Unlock service code
8. Send Store Service Code Confirmation
9. Disconnect Connection

FILENAME PROMONET_DCM_USECASES_12.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 5/8/2012
	PAGE 17 OF 26	REVISED 3/12/2013



## Retrieve Configuration Update Package

### Narrative Description:

Local service employee wants to check if there is any configuration update pending in update task list. Product is in the field beyond straight mobile communication network coverage. Using service terminal the employee establishes a secure mobile communication connection to the DCM system. Then the employee sends "Check Update" message to the DCM system. DCM System identifies and authenticates the employee, and if the employee is authorized for the Check Update request and an update is pending, retrieves the current Service Code, locks it and writes one of the Configuration Update Tasks in the Configuration Update Task List to mobile storage in a Configuration Update Package using the service terminal. DCM system sends "Check Update Confirmation" and closes the connection.

### Description:

Local service requests a configuration update and task service code for the next service to be executed for a product on field.

### Actors:

Local Service Employee  
MOL Data Management System

### Preconditions/Assumptions:

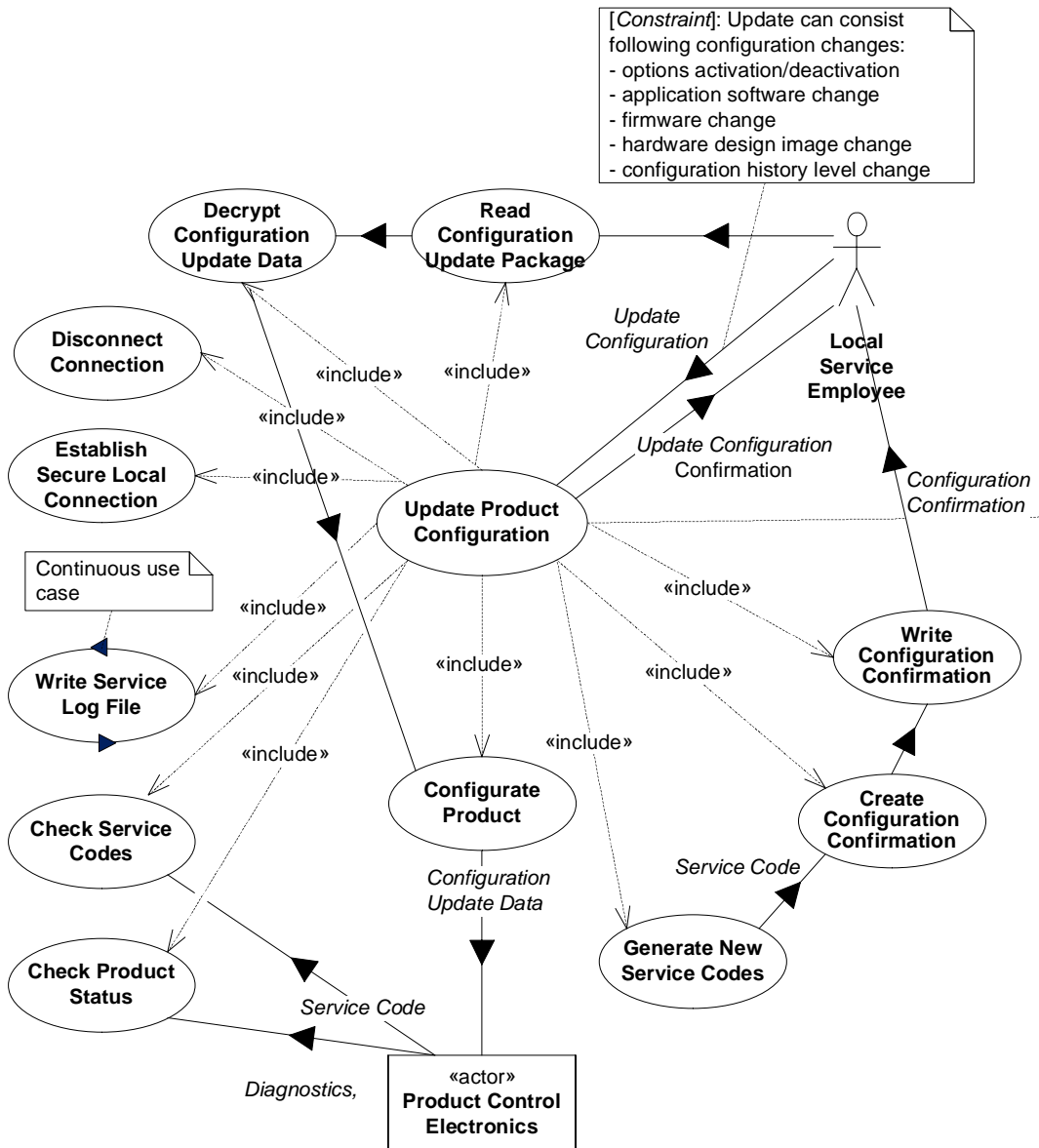
Local service has a service terminal, which is equipped with a mobile communication interface (e.g. cellular radio, satellite, or LTE), and which has memory and other resources enough to run a communication application. Service terminal also has memory to store service codes and configuration update task. That storage may be integrated to the service terminal if terminal is mobile, otherwise the storage is detachable and mobile such as USB memory stick. Configuration update data with service code is encrypted.

### Essence:

1. Establish secure mobile connection
2. Send Check Update
3. Identify and authenticate employee
4. Verify authorization
5. Lock service code
6. Create Configuration Update Package
7. Write Configuration Update Package to mobile storage
8. Send Check Update Confirmation
9. Disconnect Connection

FILENAME	DRAWN BY	DATE
PROMONET_DCM_USECASES_12.VSD	PARKKILA TOMMI, ISTO PEKKA	5/8/2012
	PAGE	REVISED
	18 OF 26	3/12/2013





## Update Product Configuration

*Narrative Description:*

Local service employee wants to update product configuration. The employee establishes a secure local communication connection to the product. Then the employee uses service terminal to send "Update Configuration" message to the product for reading service code and configuration update package from mobile storage. Product checks that the service code matches with the current service code stored into local memory of the product. If the service code is valid, product check its status for executing the configuration update, and updates configuration with new configuration settings from Configuration Update Package when possible. After succesful configuration update and storing, product generates new service code, writes Configuration Confirmation including new service code to mobile storage. DCM system indicates completion with "Update Configuration Confirmation" and closes the connection.

*Description:*

Local service updates product configuration.

*Actors:*

Local Service Employee

Product Control Electronics

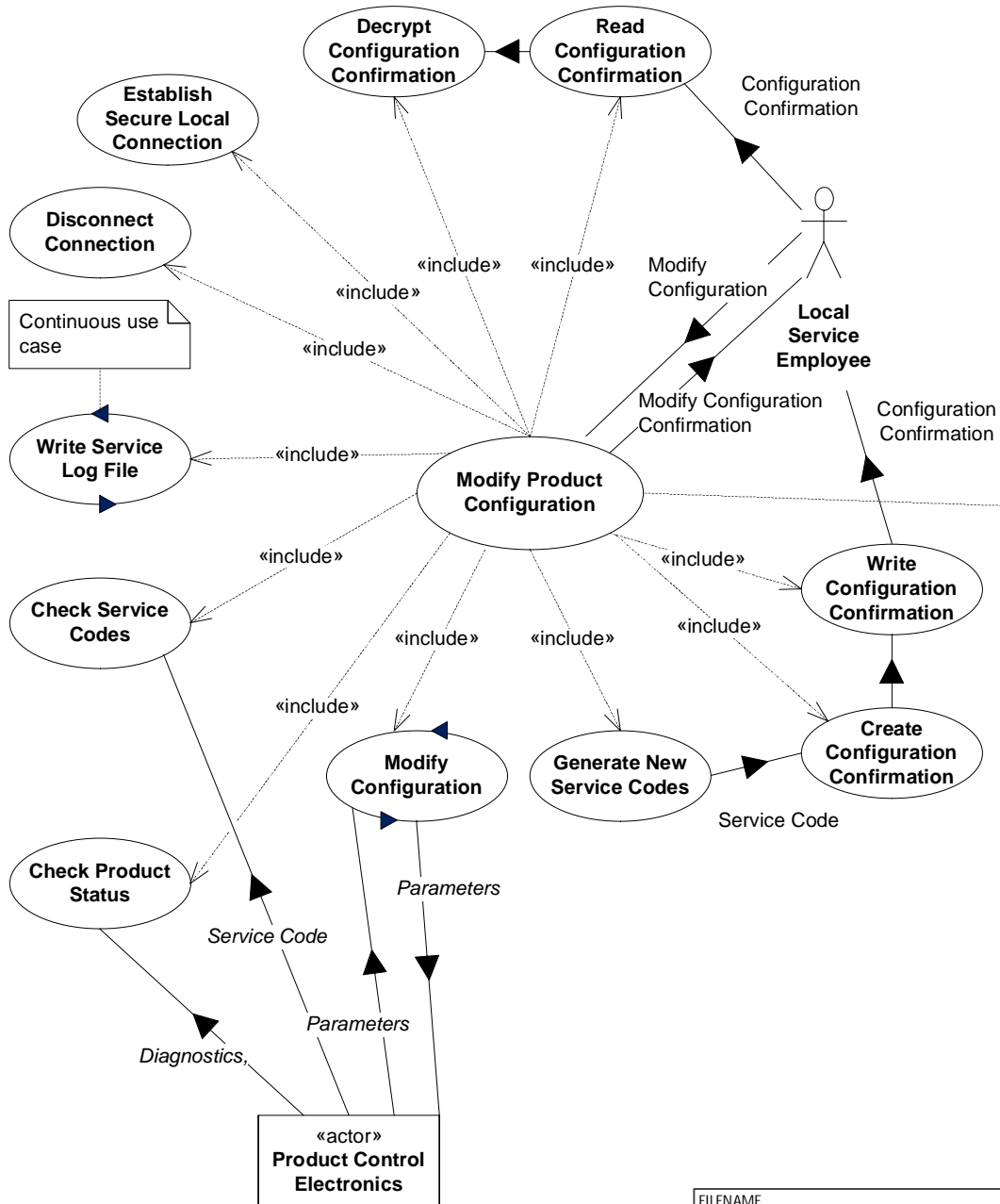
*Preconditions/Assymptions:*

Local service has a service terminal, which is equipped with a local communication interface (e.g. Short range radio, or USB), and which has memory and other resources enough to run a communication application and storage for product configuration data. The storage may be detachable and mobile. Product has a local communication interface as well or the service terminal is integrated to the product and mobile storage is connected to it at the time of update. Service code pair consist of unique product ID number and random service code (PIN), which is updated by the product after every successful configuration update.

*Essence:*

1. Establish secure local connection
2. Send Update Configuration
3. Read Configuration Update Package from mobile storage
4. Decrypt Configuration Update Package
5. Check service codes
6. Check product status
7. Configure product
8. Generate new service codes
9. Create Configuration Confirmation
10. Write Update Configuration Confirmation to Mobile Storage
11. Send Update Configuration Confirmation
12. Disconnect Connection

FILENAME	PROMONET_DCM_USECASES_12.VSD	DRAWN BY	PARKKILA TOMMI, ISTO PEKKA	DATE	5/8/2012
		PAGE	19 OF 26	REVISED	3/12/2013



## Modify Product Configuration

### Narrative Description:

Local service employee wants to modify product configuration data with a user interface of a terminal device. Employee establish a secure local connection to product, which after employee sends "Modify Configuration" request. The terminal reads encrypted Configuration Confirmation from mobile storage and transfers it to the product. Product checks the match of the service code in the Configuration Confirmation and when safe shows parameter values of the current configuration to the employee for editing. The employee changes parameter values of the current configuration and when done, product updates Configuration Update Data, creates new encrypted Configuration Confirmation packet with the new Service Code and transfers it to the terminal which writes it to the mobile storage. DCM system indicates completion with "Modify Configuration Confirmation" and closes the connection.

### Description:

Local service modifies configuration data in product on field.

### Actors:

Local Service Employee  
Product Control Electronics

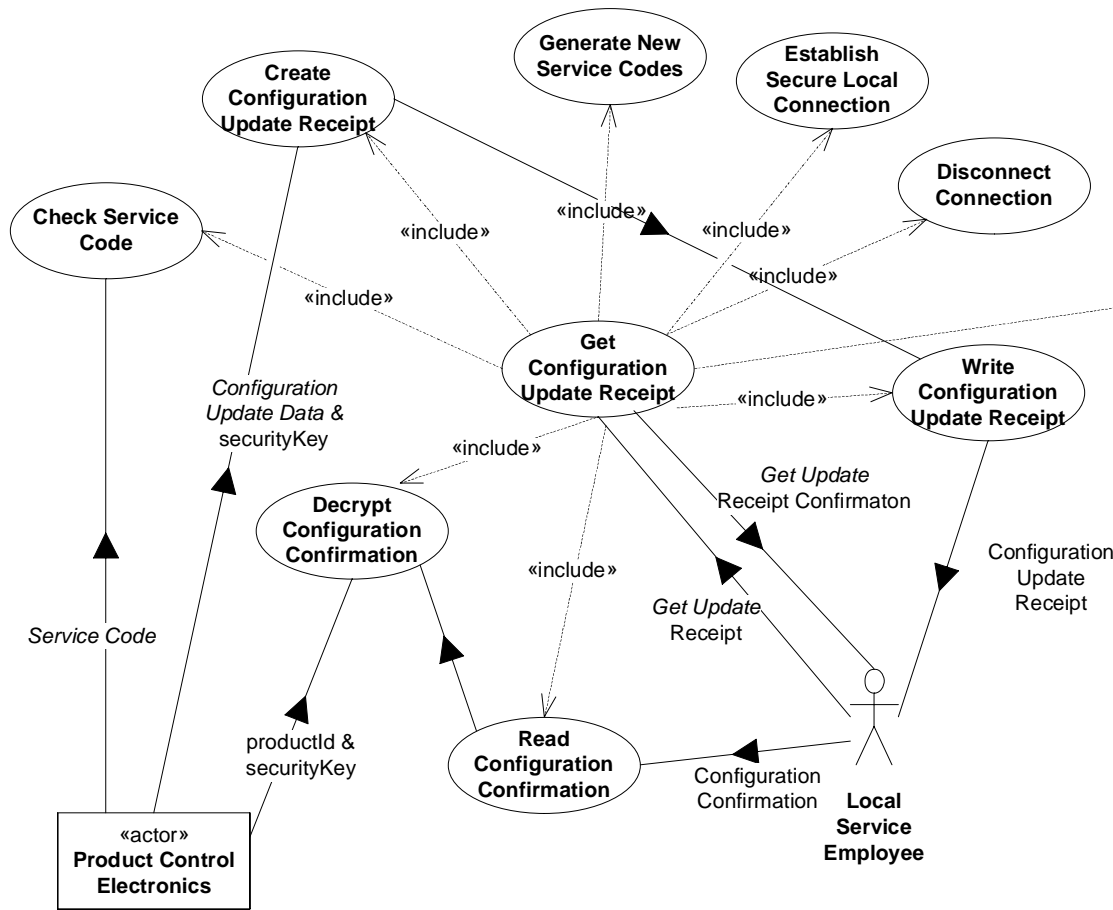
### Preconditions/Assumptions:

See use case Update Product Configuration.

### Essence:

1. Establish secure local connection
2. Send Modify Configuraton
3. Read Configuration Confirmation from mobile storage
4. Decrypt Configuration Confirmation
5. Check service codes
6. Check product status
7. Modify Product Configuration and update Configuration Update Data
8. Generate new service codes
9. Update service code in Configuration Update Data
10. Update service code in Configuration Confirmation
11. Create Configuration Confirmation
12. Write Configuration Confirmation
13. Send Modify Configuraton Confirmation
14. Disconnect Connection

FILENAME	DRAWN BY	DATE
PROMONET_DCM_USECASES_12.VSD	PARKKILA TOMMI, ISTO PEKKA	5/8/2012
	PAGE	REVISED
	20 OF 26	3/12/2013



### Get Configuration Update Receipt

*Narrative Description:*  
 Local service employee wants to get a receipt of executed configuration service done for the product. Employee sends "Get Update Receipt" request to product. The terminal reads encrypted Configuration Confirmation from mobile storage and transfers it to the product, which after, product checks the validity of the request and forms secured configuration update receipt including current service code, all configuration modifications done in the service and updated service log file and writes it to mobile storage. DCM system indicates completion with "Get Update Receipt Confirmation" and closes the connection.

*Description:*  
 Local service gets the Configuration Update Receipt from the product.

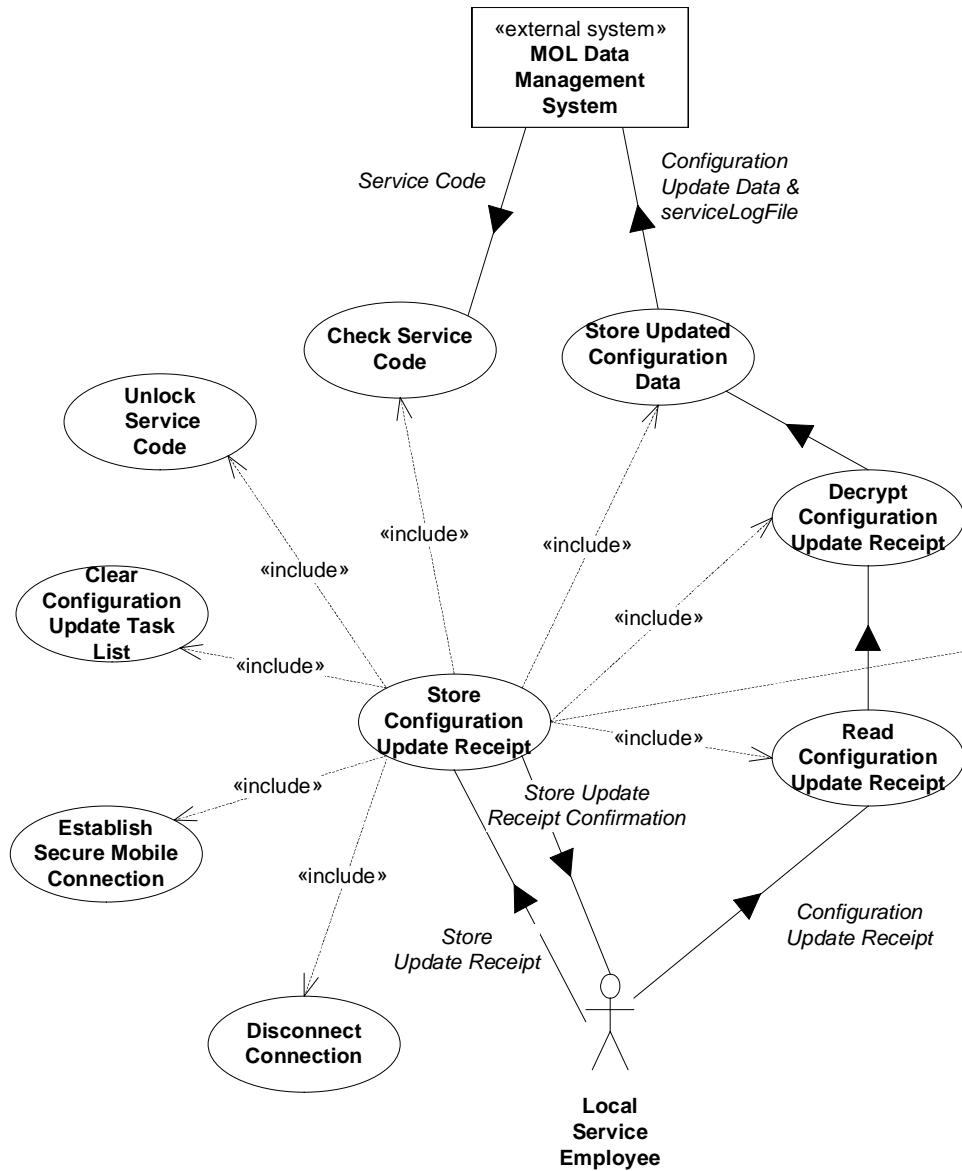
*Actors:*  
 Local Service Employee  
 Product Control Electronics

*Preconditions/Assumptions:*  
 Local service has a service terminal, which is equipped with a local communication interface (e.g. Short range radio, or USB), and which has memory and other resources enough to run a communication application and storage for product configuration data. The storage may be detachable and mobile. Product has a local communication interface as well or the service terminal is integrated to the product and mobile storage is connected to it at the time of update.

*Essence:*

1. Establish secure local connection
2. Send Get Update Receipt
3. Read Configuration Confirmation from mobile storage
4. Decrypt Configuration Confirmation
5. Check service code in Configuration Confirmation
6. Generate new service code
7. Create Configuration Update Receipt
8. Write receipt to mobile storage
9. Send Get Update Receipt Confirmation
10. Disconnect Connection

FILENAME	DRAWN BY	DATE
PROMONET_DCM_USECASES_12.VSD	PARKKILA TOMMI, ISTO PEKKA	5/8/2012
	PAGE	REVISED
	21 OF 26	3/12/2013



### Store Configuration Update Receipt

*Narrative Description:*

Local service employee wants to upload and update current configuration data of a product to MOL Data Management System. Local service establish a secure mobile communication connection to the DCM system with service terminal. Then The employee sends "Store Update Receipt" message to DCM system. The service terminal reads Configuration update Receipt from the mobile storage and sends it to DCM system. DCM system receives and decrypts it and checks that the old service code stored in the Configuration Update Data and MOL Data Management System match. DCM system stores configuration data to MOL Data Management system, clears the task from update task list, unlocks the service code, and returns "Store Update Receipt Confirmation" message to local service and service terminal closes the connection.

*Description:*

Local service updates product configuration data stored into MOL Data Management System with current configuration from product on field.

*Actors:*

- Local Service Employee
- MOL Data Management System

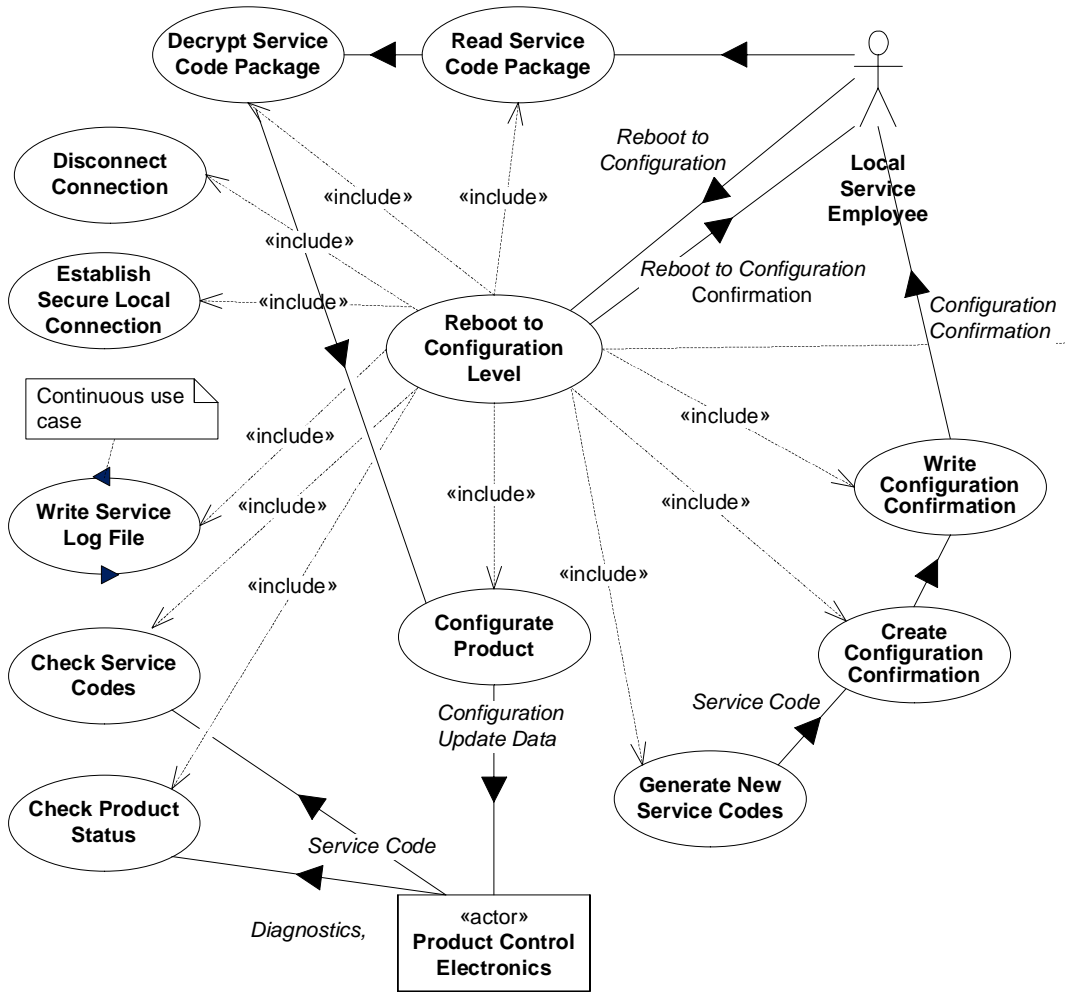
*Preconditions/Assumptions:*

Local service has a service terminal, which is equipped with a mobile communication interface (e.g. cellular radio, satellite, or LTE), and which has memory and other resources enough to run a communication application. Service terminal also has memory to store service codes and configuration update receipt. That storage may be integrated to the service terminal if terminal is mobile, otherwise the storage is detachable and mobile such as USB memory stick.

*Essence:*

1. Establish secure mobile connection
2. Send Store Update Receipt
3. Read Configuration Update Receipt from mobile storage
4. Decrypt Configuration Update Receipt
5. Check service codes
6. Store Updated Configuration Data to MOL Data Management System
7. Clear task from Configuration Update Task List
8. Unlock service code
9. Send Store Update Receipt Confirmation
10. Disconnect Connection

FILENAME PROMONET_DCM_USECASES_12.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 5/8/2012
	PAGE 22 OF 26	REVISED 3/12/2013



## Reboot to Configuration Level

### Narrative Description:

Local service employee wants to revert to one of configurations stored in the product: Factory reset configuration, previous configuration or newest configuration. The employee establishes a secure local communication connection to the product and uses service terminal to send "Reboot to Configuration" message to the product for reading service code from mobile storage. Product checks that the service code matches with the product's current service code. If the service code is valid, product check its status for executing the configuration revert and performs it when possible. After succesful revert, product generates new service code, writes Configuration Confirmation including new service code to mobile storage. DCM system indicates completion with "Revert to Configuration Confirmation" and closes the connection.

### Description:

Local service reverts product configuration.

### Actors:

Local Service Employee

Product Control Electronics

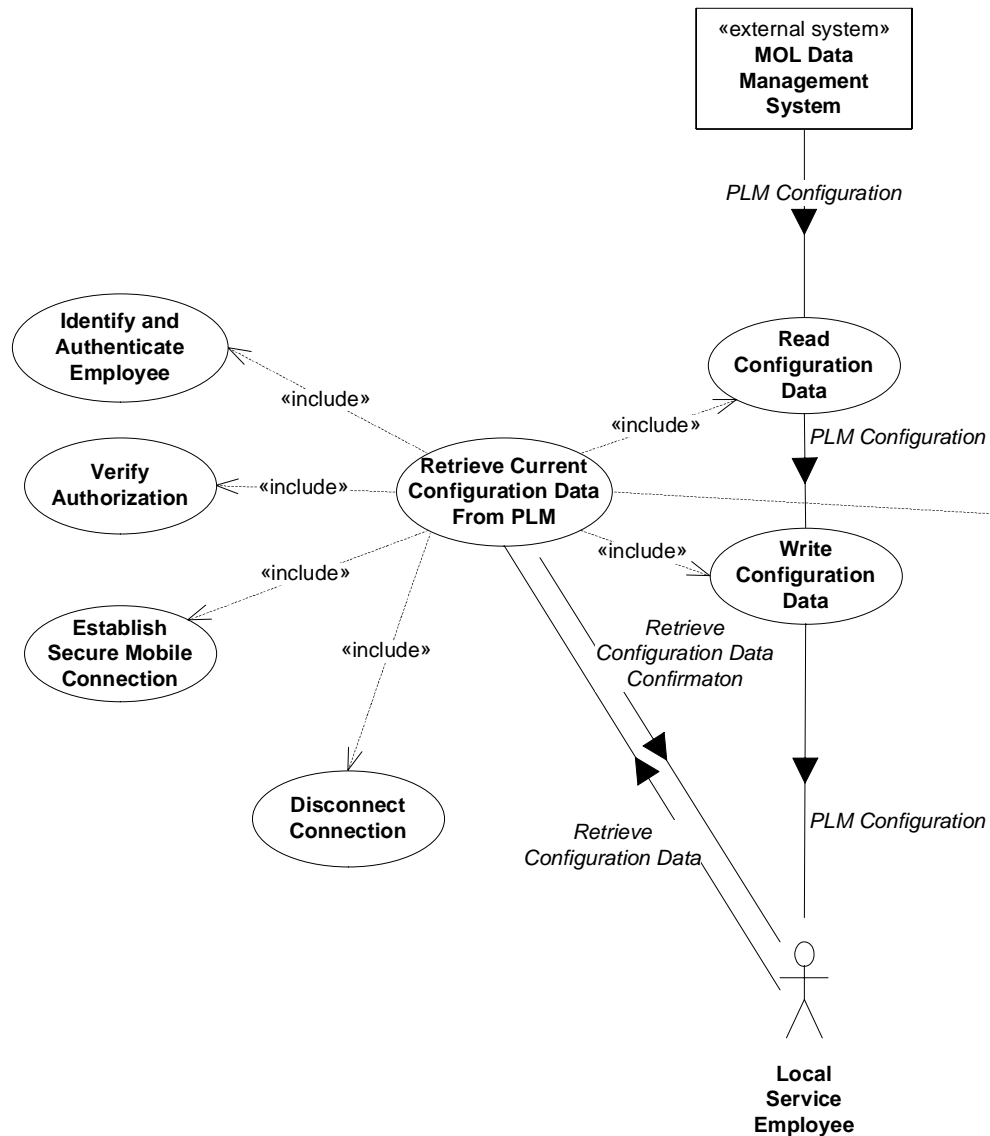
### Preconditions/Assumptions:

Local service has a service terminal, which is equipped with a local communication interface (e.g. Short range radio, or USB), and which has memory and other resources enough to run a communication application and storage for product configuration data. The storage may be detachable and mobile. Product has a local communication interface as well or the service terminal is integrated to the product and mobile storage is connected to it at the time of update. Service code pair consist of unique product ID number and random service code (PIN), which is updated by the product after every successful configuration update.

### Essence:

1. Establish secure local connection
2. Send Reboot to Configuration
3. Read Service Code Package from mobile storage
4. Decrypt Service Code Package
5. Check service codes
6. Check product status
7. Configure product
8. Generate new service codes
9. Create Configuration Confirmation
10. Write Update Configuration Confirmation to Mobile Storage
11. Send Update Configuration Confirmation
12. Disconnect Connection

FILENAME	PROMONET_DCM_USECASES_12.VSD	DRAWN BY	PARKKILA TOMMI, ISTO PEKKA	DATE	5/8/2012
		PAGE	23 OF 26	REVISED	3/12/2013



### Retrieve Current Configuration Data From PLM

*Narrative Description:*

Local service employee wants to download current configuration data of a product from MOL Data Management System using the service terminal. The employee establishes a secure mobile communication connection to the DCM system. Then the employee sends "Retrieve Configuration Data" request to the DCM system. DCM system authenticates the local service, checks which product data the employee is authorized to get access rights and if authorized reads corresponding product specific configuration data from MOL Data Management System and returns the configuration data. Service terminal writes the configuration data to the mobile storage, DCM system indicates completion with "Retrieve Configuration Data Confirmation". Service terminal closes the connection.

*Description:*

Local service requests current configuration data of a specific product from PLM.

*Actors:*

- Local Service Employee
- MOL Data Management System

*Preconditions/Assumptions:*

Local service has a service terminal, which is equipped with a mobile communication interface (e.g. cellular radio, satellite, or LTE), and which has memory and other resources enough to run a communication application. Service terminal also has memory to store service codes and configuration update task list. That storage may be integrated to the service terminal if terminal is mobile, otherwise the storage is detachable and mobile such as USB memory stick.

*Essence:*

1. Establish secure mobile connection
2. Send Retrieve Configuration Data
3. Identify and authenticate employee
4. Verify authorization
5. Read configuration data from MOL Data Management System
6. Write configuration data to mobile storage
7. Send Retrieve Configuration Data Confirmation
8. Disconnect Connection

FILENAME PROMONET_DCM_USECASES_12.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 5/8/2012
	PAGE 24 OF 26	REVISED 3/12/2013

### Get Current Configuration Data From Product

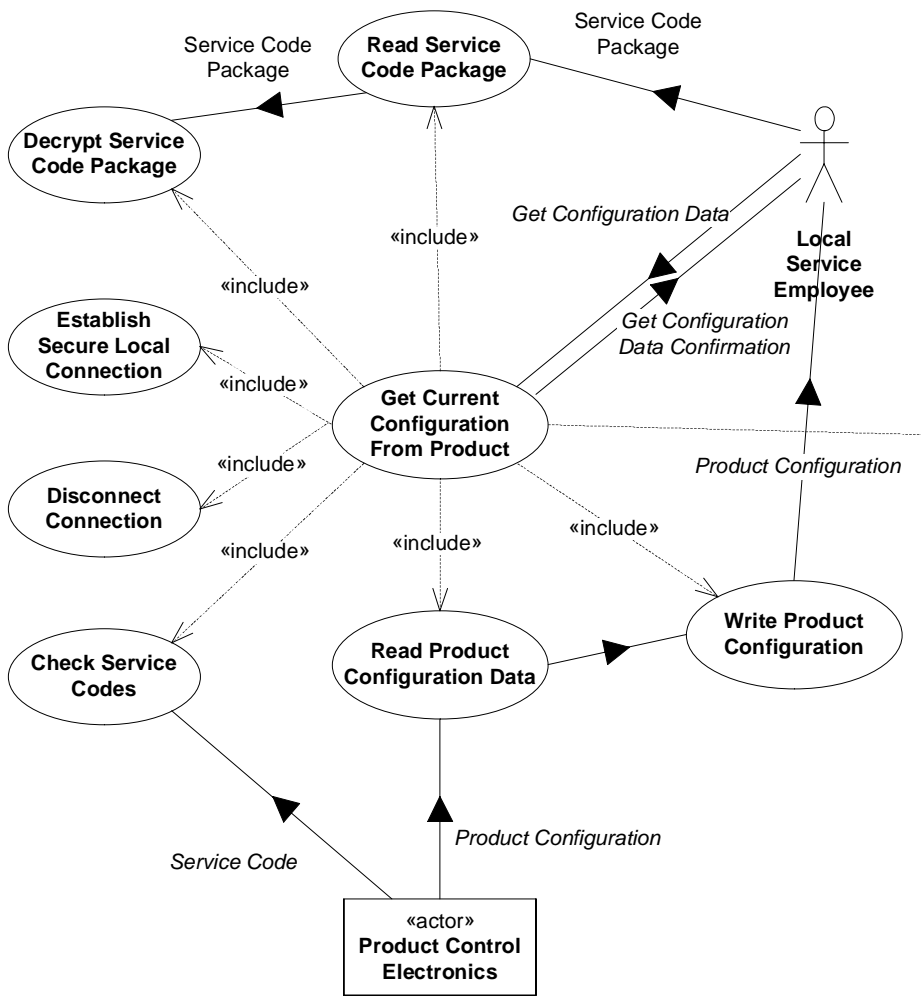
**Narrative Description:**  
 Local service employee wants to get product configuration data. The employee establishes a secure local communication connection to the product. Then The employee sends "Get Configuration Data" request to the Product using service terminal. The product reads Service Code Package from the mobile storage, decrypts it and compares the service code to current one in the product. If the codes match the product writes current product configuration to mobile storage and indicates completion with "Get Configuration Data Confirmation". Local service closes the connection.

**Description:**  
 Local service requests configuration data from product

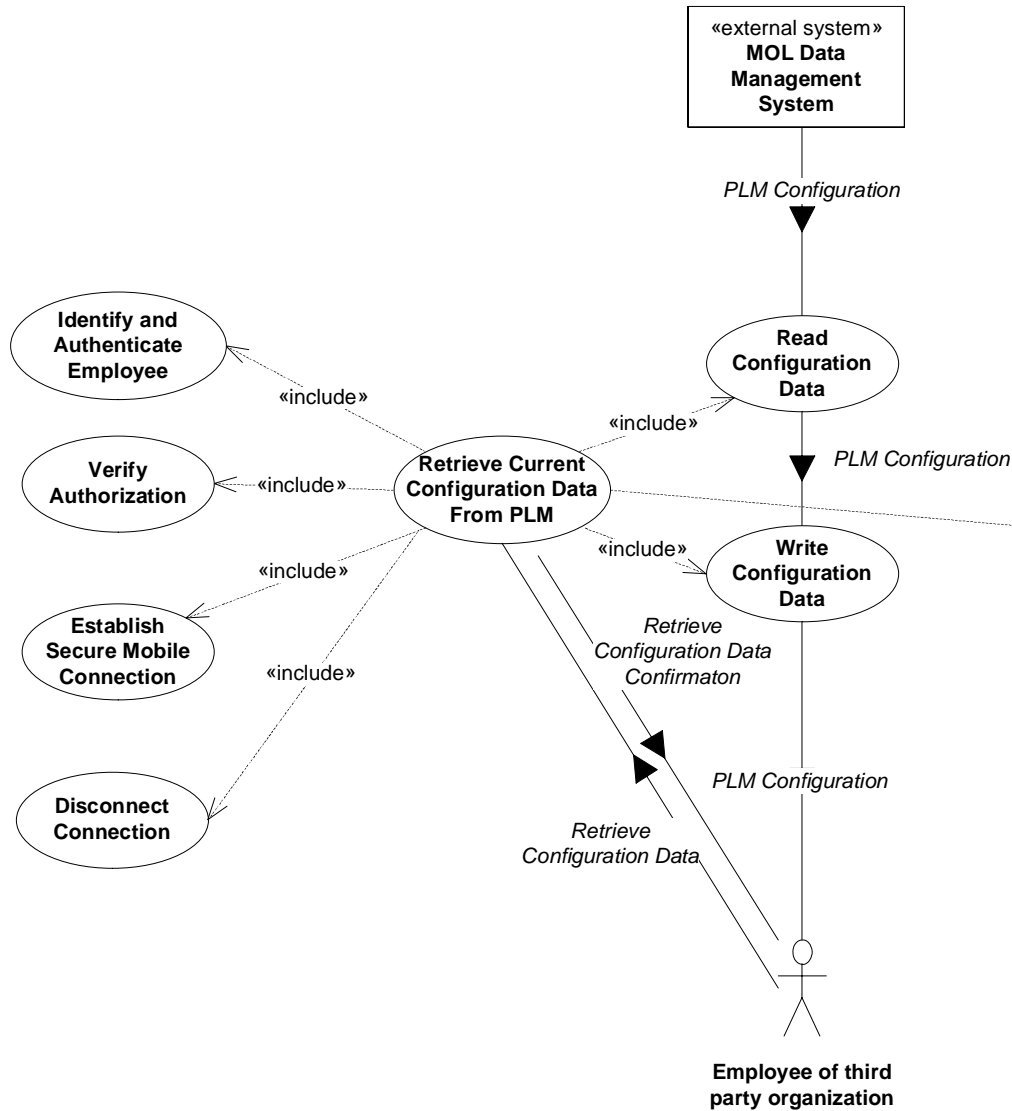
**Actors:**  
 Local Service Employee  
 Product Control Electronics

**Preconditions/Assumptions:**  
 Local service has a service terminal, which is equipped with a local communication interface (e.g. Short range radio, or USB), and which has memory and other resources enough to run a communication application and storage for product configuration data. The storage may be detachable and mobile. Product has a local communication interface as well or the service terminal is integrated to the product and mobile storage is connected to it at the time of update. The local service employee has retrieved current service code for the product in Service Code Package on the mobile storage.

- Essence:**
1. Establish secure local connection
  2. Send Get Configuration Data
  3. Read Service code Package from mobile storage
  4. Decrypt Service code Package
  5. Check service codes
  6. Read product configuration data
  7. Write configuration data to mobile storage
  8. Send Get Configuration Data Confirmation
  9. Disconnect Connection



FILENAME	DRAWN BY	DATE
PROMONET_DCM_USECASES_12.VSD	PARKKILA TOMMI, ISTO PEKKA	5/8/2012
	PAGE	REVISED
	25 OF 26	3/12/2013



### Retrieve Current Configuration From PLM

*Narrative Description:*

Third party employee wants to download current configuration data of their sub-device of a product from MOL Data Management System. Local service employee wants to download current configuration data of a product from MOL Data Management System. The third party employee establishes a secure mobile communication connection to the DCM system. Then the third party employee sends "Retrieve Configuration Data" request to the DCM system. DCM system authenticates the third party employee, checks which device data on the product the third party employee is authorized to get access rights and if authorized reads corresponding product specific configuration data for the device from MOL Data Management System and returns it. Service terminal writes the configuration data to the mobile storage, DCM system indicates completion with "Retrieve Configuration Data Confirmation". Service terminal closes the connection.

*Description:*

Third party employee requests current configuration data of a specific sub-device of a product.

*Actors:*

Employee of third party organization  
MOL Data Management System

*Preconditions/Assymptions:*

Third party employee has a service terminal, which has memory and other resources enough to run a communication application and store product configuration data. Service terminal also has memory to store service codes and configuration data. That storage may be integrated to the service terminal if terminal is mobile, otherwise the storage is detachable and mobile such as USB memory stick.

*Essence:*

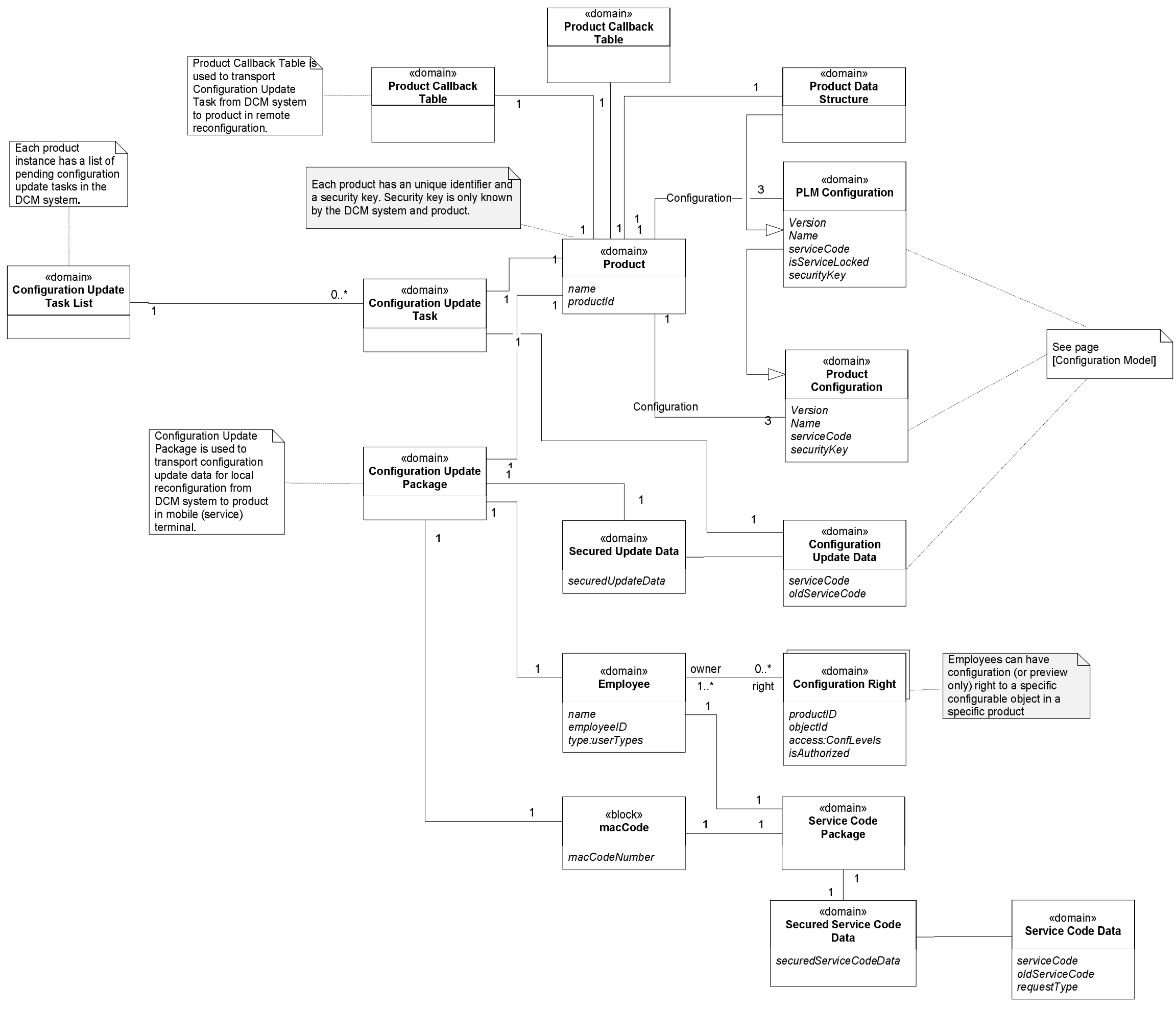
1. Establish secure mobile connection
2. Send Retrieve Configuration Data
3. Identify and authenticate employee
4. Verify authorization
5. Read device configuration data from MOL Data Management System
6. Write configuration data to mobile storage
7. Send Retrieve Configuration Data Confirmation
8. Disconnect Connection

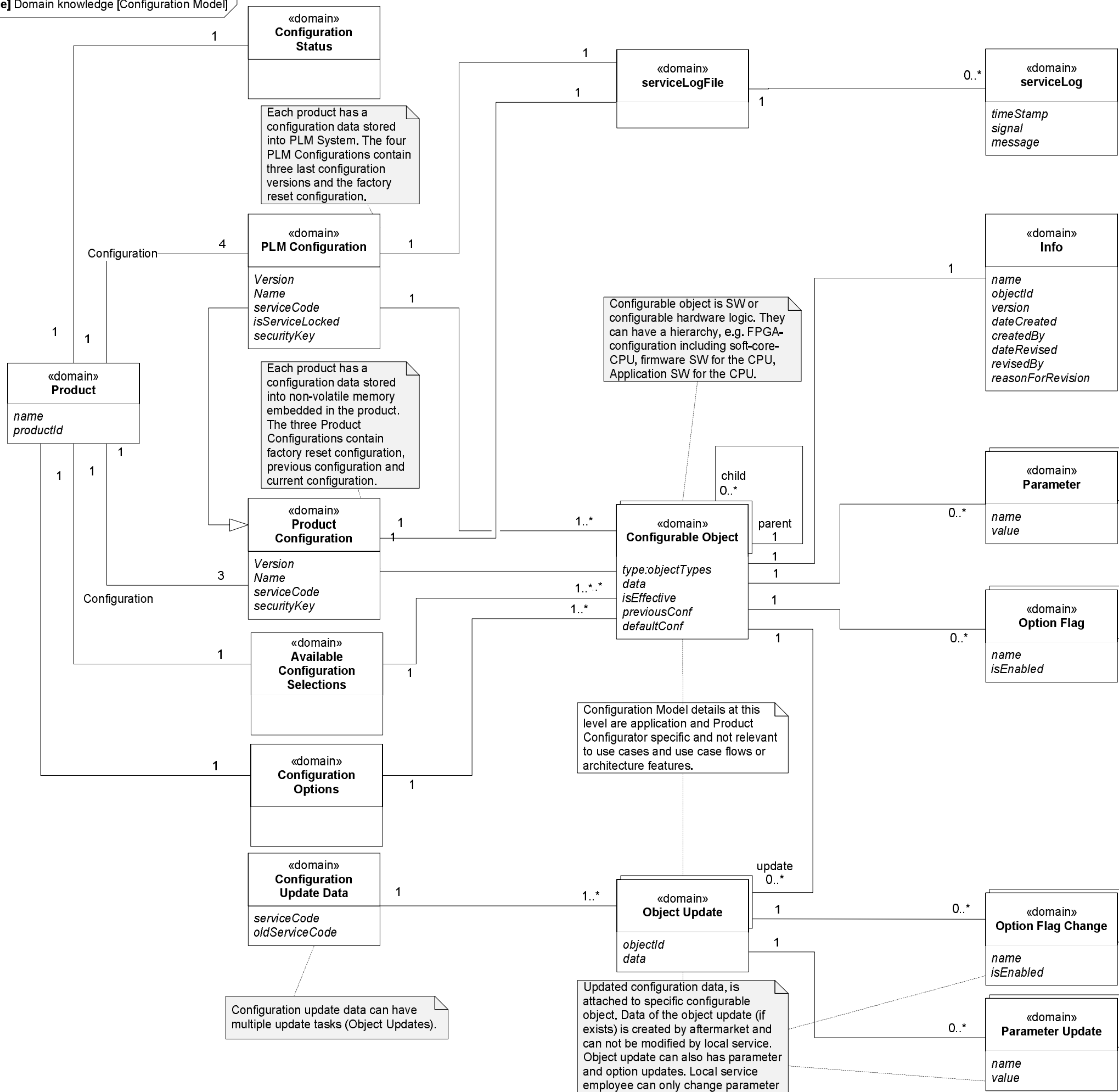
FILENAME PROMONET_DCM_USECASES_12.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 5/8/2012
	PAGE 26 OF 26	REVISED 3/12/2013



TITLE	ProMoNet DCM Domain Knowledge		DESCRIPTION	Domain knowledge diagrams of ProMoNet dynamic configuration management (DCM) system for reconfigurable networked industrial products	
FILENAME	PROMONET_DCM_DOMAIN KNOWLEDGE_09.VSD		DRAWN BY	PARKKILA TOMMI, ISTO PEKKA	
			DATE	9/11/2012	
			PAGE	1 OF 6	
			REVISED	4/21/2013	

REVISIONS			
REV.	DESCRIPTION	DATE	BY
0.1	First draft created	05/15/2012	TOP
0.2	Enumerations added, requests added	09/25/2012	TOP
0.3	Added Configuration Update Package, reorganizations and minor updates	11/12/2012	PEI
0.4	Added Secured Update Data and Secured Receipt.	11/13/2012	PEI
0.5	Conflicting revisions from TOP and PEI – Revision deleted.	--	--
0.6	Merged changes of 0.5 by TOP (review) and 0.5 by PEI (use cases 0.7).	11/23/2012	PEI
0.7	Separated configuration model. Updated to flows 0.9.	02/28/2013	PEI
0.8	Corrected for review findings: Local reboot and versioning.	03/12/2013	PEI
0.9	Added comments.	04/21/2013	PEI

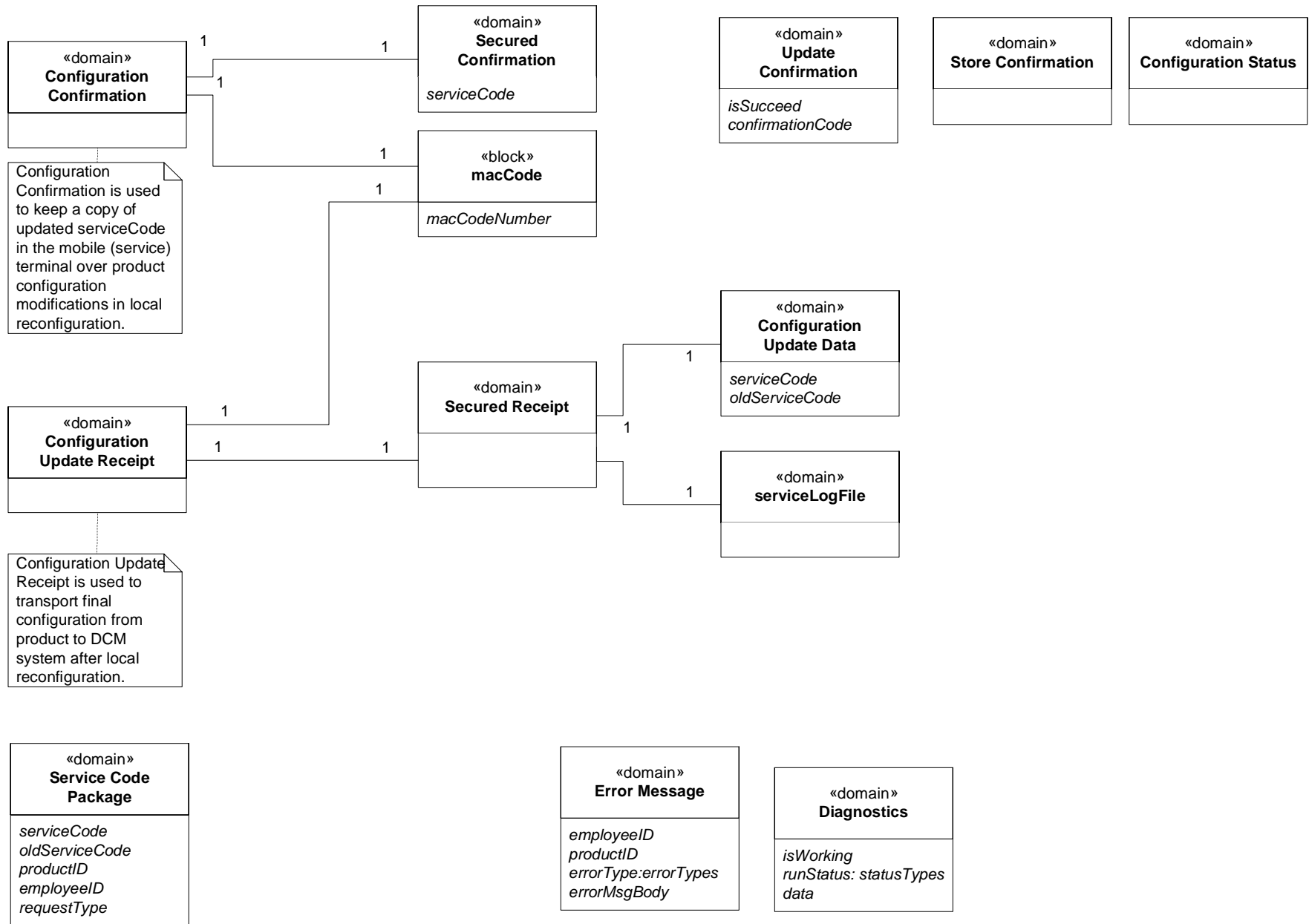




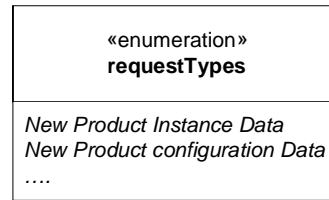
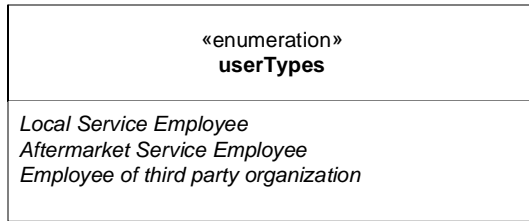
Configuration update data can have multiple update tasks (Object Updates).

Updated configuration data, is attached to specific configurable object. Data of the object update (if exists) is created by aftermarket and can not be modified by local service. Object update can also has parameter and option updates. Local service employee can only change parameter values after retrieving configuration confirmation from the product updated.

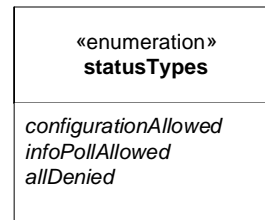
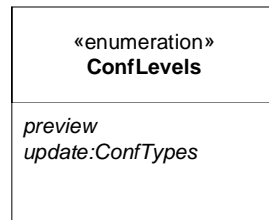
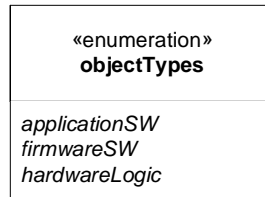
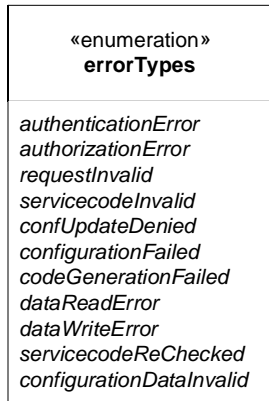
Each configurable object can have parameters (runtime and boot-time) and also options for configuring application or firmware SW. Local service can only edit parameters in the field by themselves.



FILENAME PROMONET_DCM_DOMAIN KNOWLEDGE_09.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 9/11/2012
	PAGE 4 OF 6	REVISED 4/21/2013

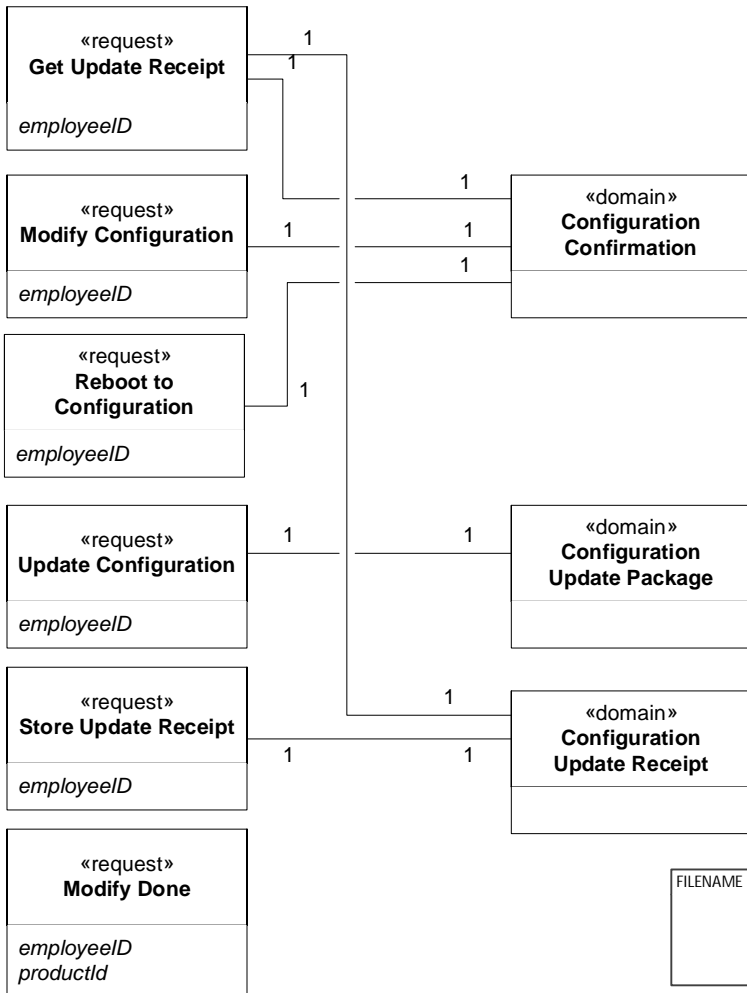
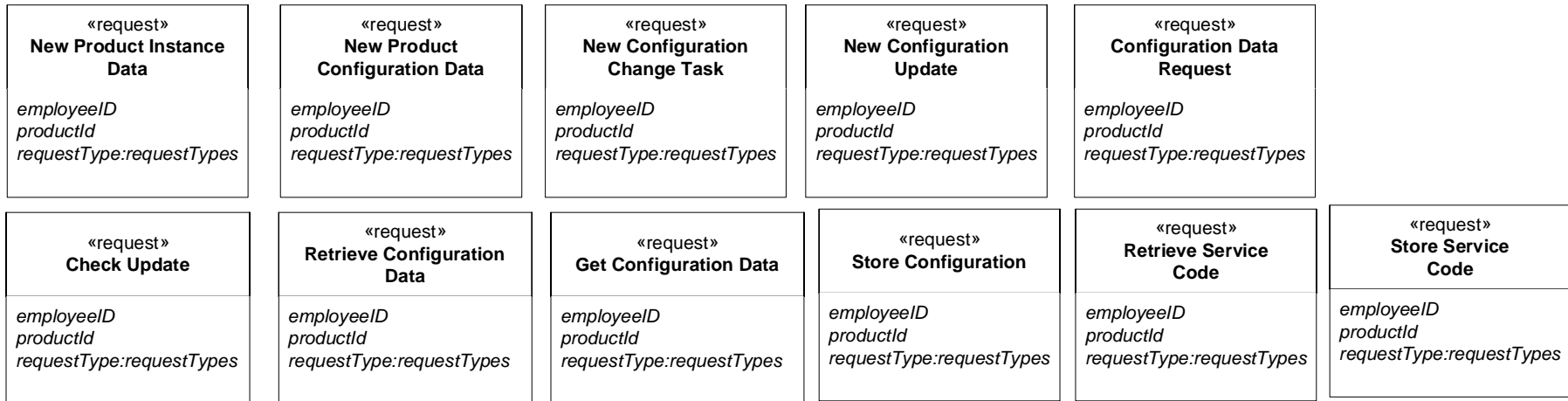


Requests to the DCM system, see page [Requests]



The value of *update* is used to generate a default list of object updates defined for each *ConfType*.

FILENAME PROMONET_DCM_DOMAIN KNOWLEDGE_09.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 9/11/2012
	PAGE 5 OF 6	REVISED 4/21/2013

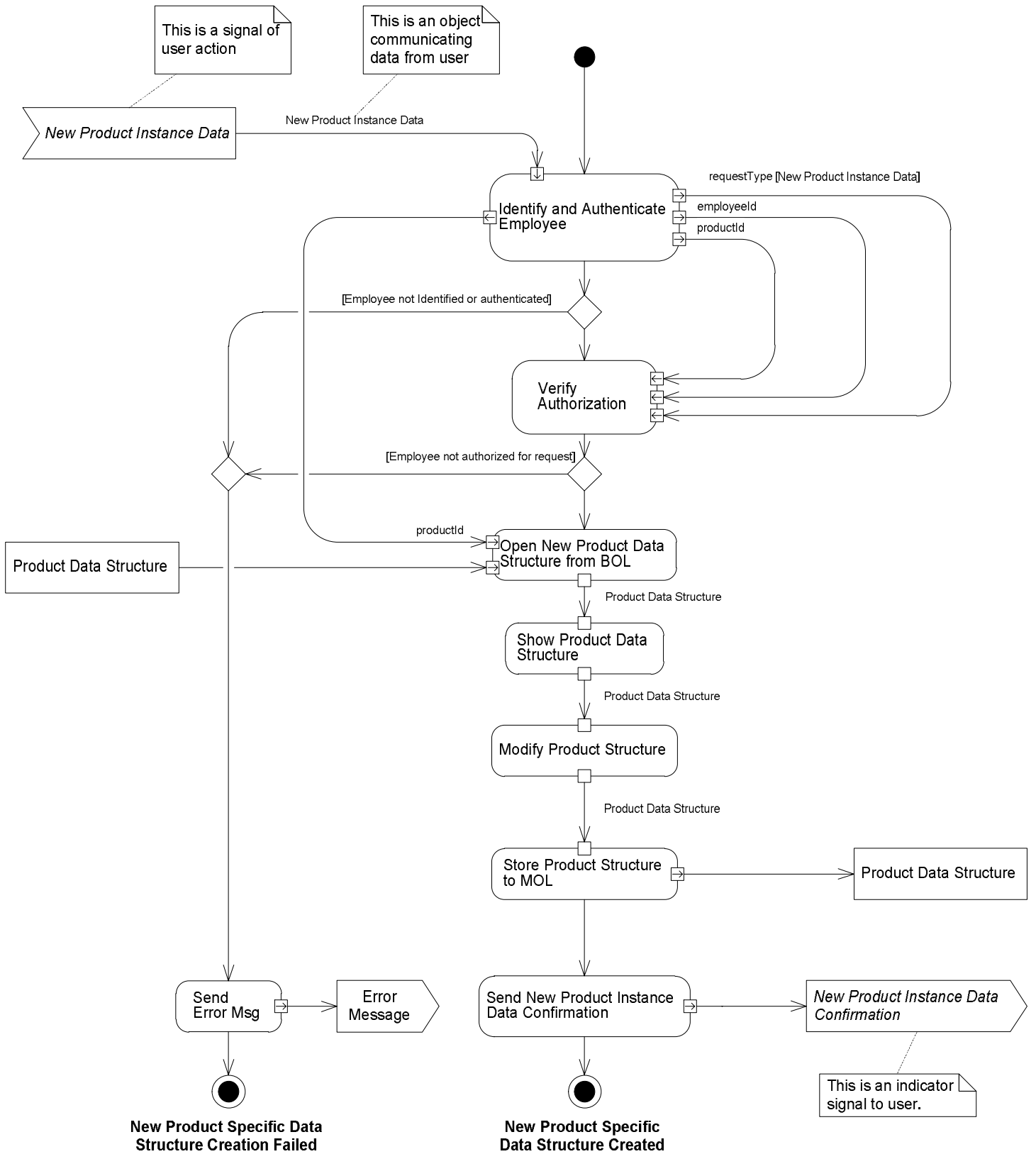


FILENAME PROMONET_DCM_DOMAIN KNOWLEDGE_09.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 9/11/2012
	PAGE 6 OF 6	REVISED 4/21/2013

TITLE	ProMoNet DCM System Use Case Flows		DESCRIPTION	Use case activity diagrams of ProMoNet dynamic configuration management (DCM) system for reconfigurable networked industrial products	
FILENAME	PROMONET_DCM_USECASEFLOWS_11.VSD	DRAWN BY	PARKKILA TOMMI, ISTO PEKKA	DATE	5/17/2012
		PAGE	1 OF 27	REVISED	5/14/2013

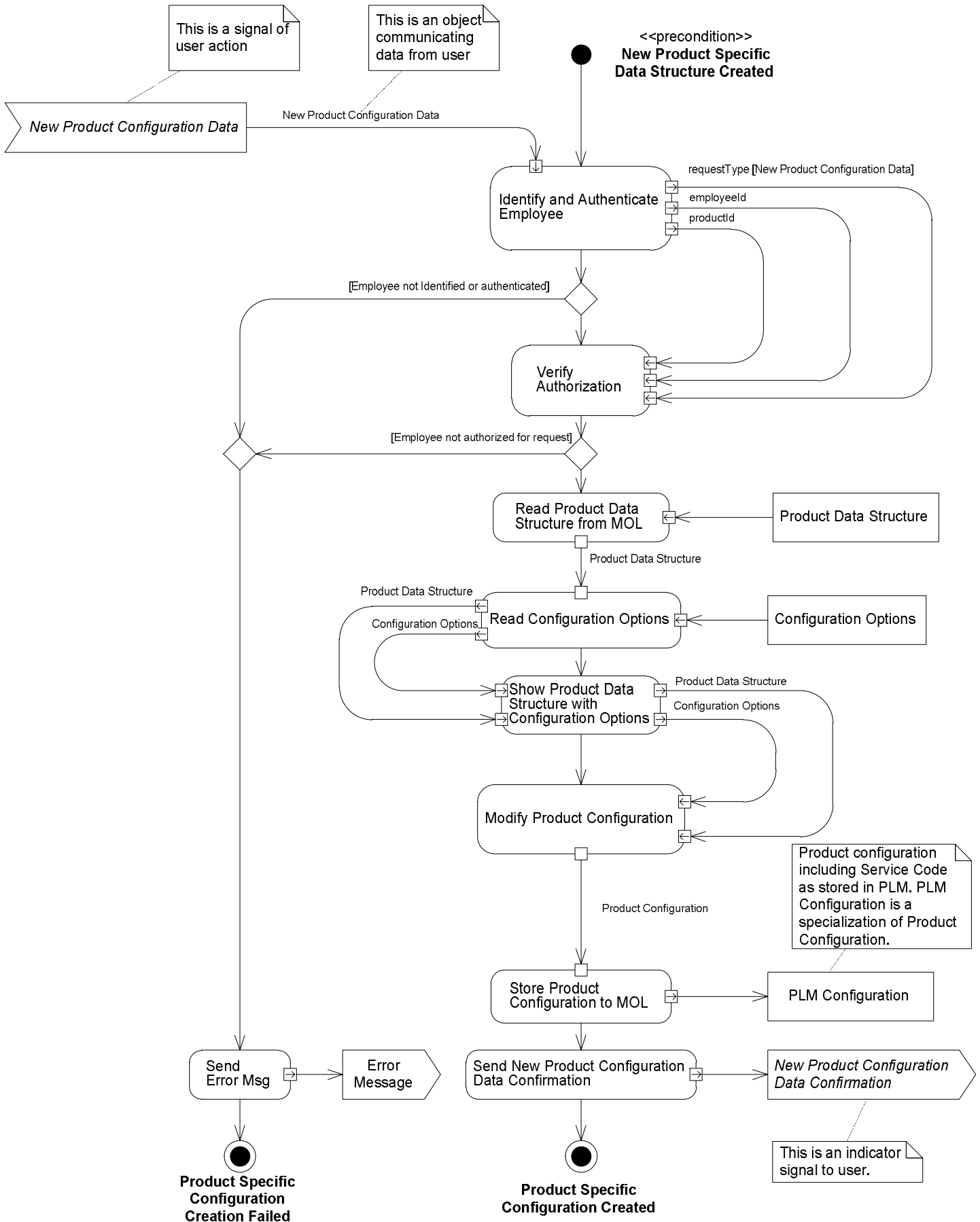
REVISIONS			
REV.	DESCRIPTION	DATE	BY
0.1	First draft created	05/17/2012	TOP
0.2	Local Service flows done	09/10/2012	TOP
0.3	Aftermarket flows added.	09/26/2012	TOP
0.4	Major modifications for local service flows	09/26/2012	TOP
0.5	Updated according to review findings	11/13/2012	PEI
0.6	Aligned to use cases rev. 0.7.	11/22/2012	PEI
0.7	Aligned to domain knowledge rev. 0.6.	11/21/2012	PEI
0.8	Added service code activities. Signal and object flows redesigned for Mobile Storage.	02/24/2013	PEI
0.9	Added Production flows. Redesigned ASE flows.	02/28/2013	PEI
10	Aligned to use cases rev. 12 and corrected architecturally flawed object flows.	03/12/2013	PEI
11	Redesigned state conditions. Added Third Party Employee flow.	05/14/2013	PEI

act Create Product Specific Data Structure [Production]

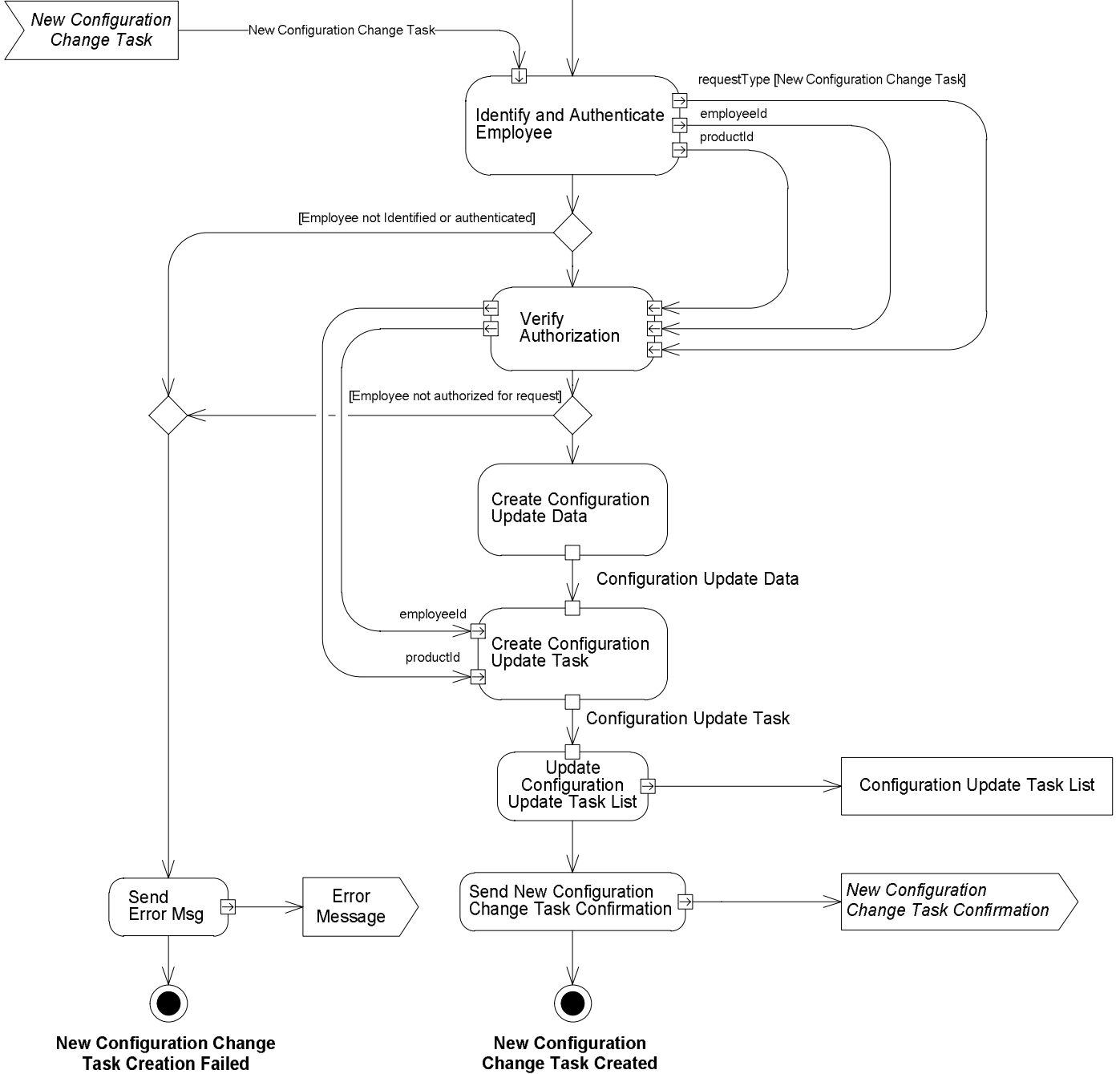


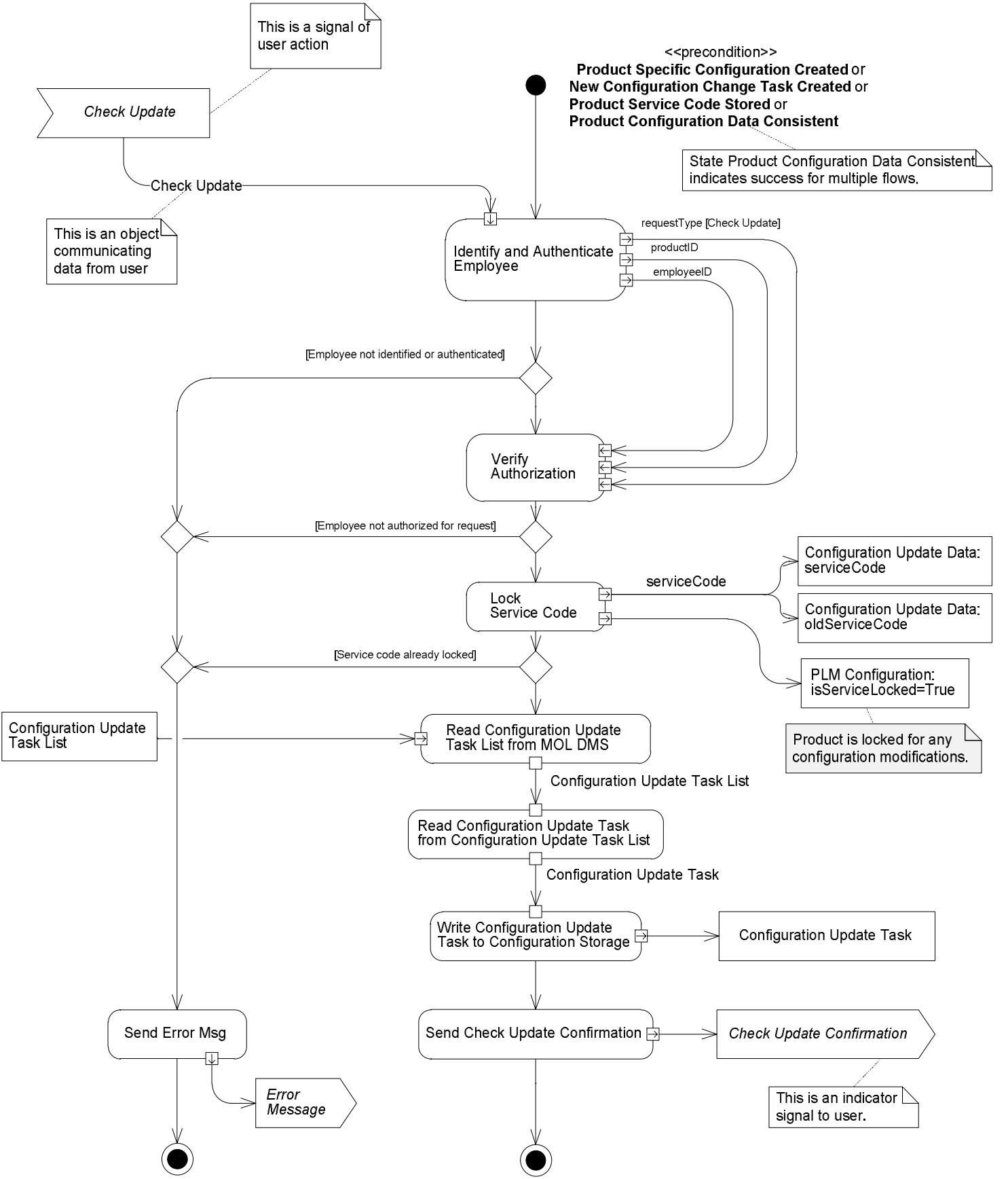
FILENAME PROMONET_DCM_USECASEFLOWS_11.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 5/17/2012
	PAGE 2 OF 27	REVISED 2/26/2013





<<precondition>>  
**Product Specific Configuration Created or  
 Product Configuration Data Consistent or  
 Product Service Code Stored**

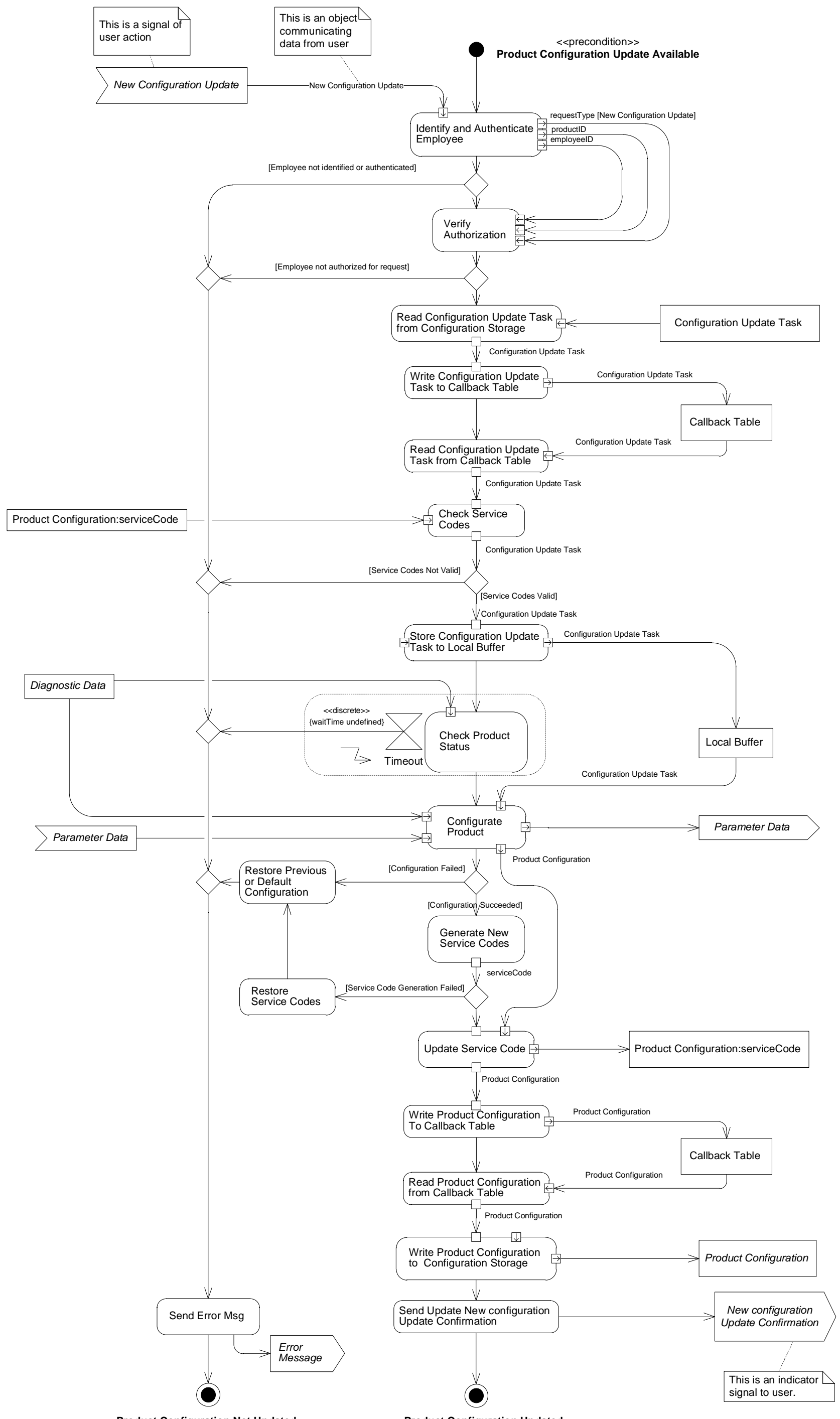




Product Configuration Update Not Available

Product Configuration Update Available

FILENAME PROMONET_DCM_USECASEFLOWS_11.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 5/17/2012
	PAGE 5 OF 27	REVISED 5/14/2013

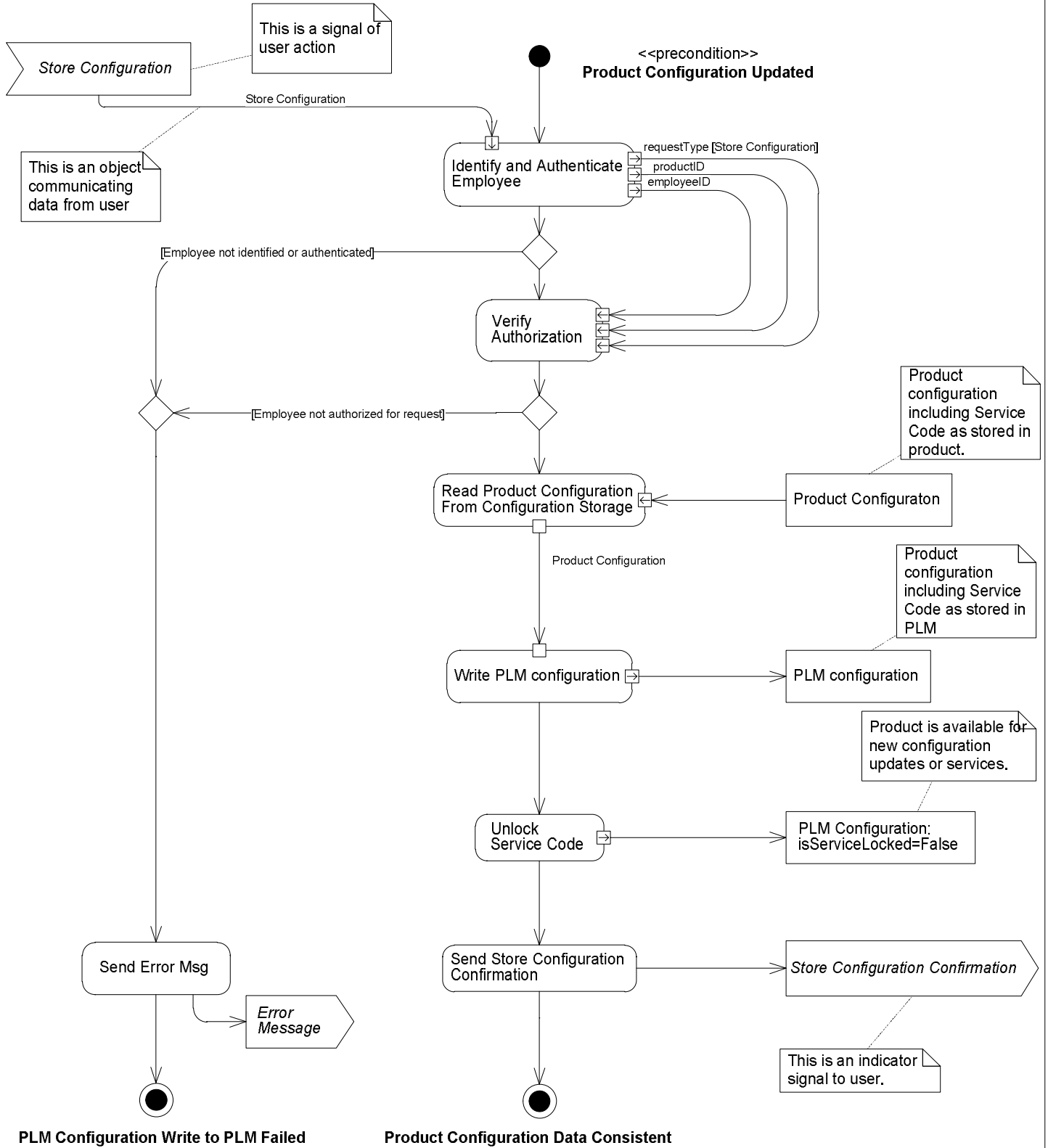


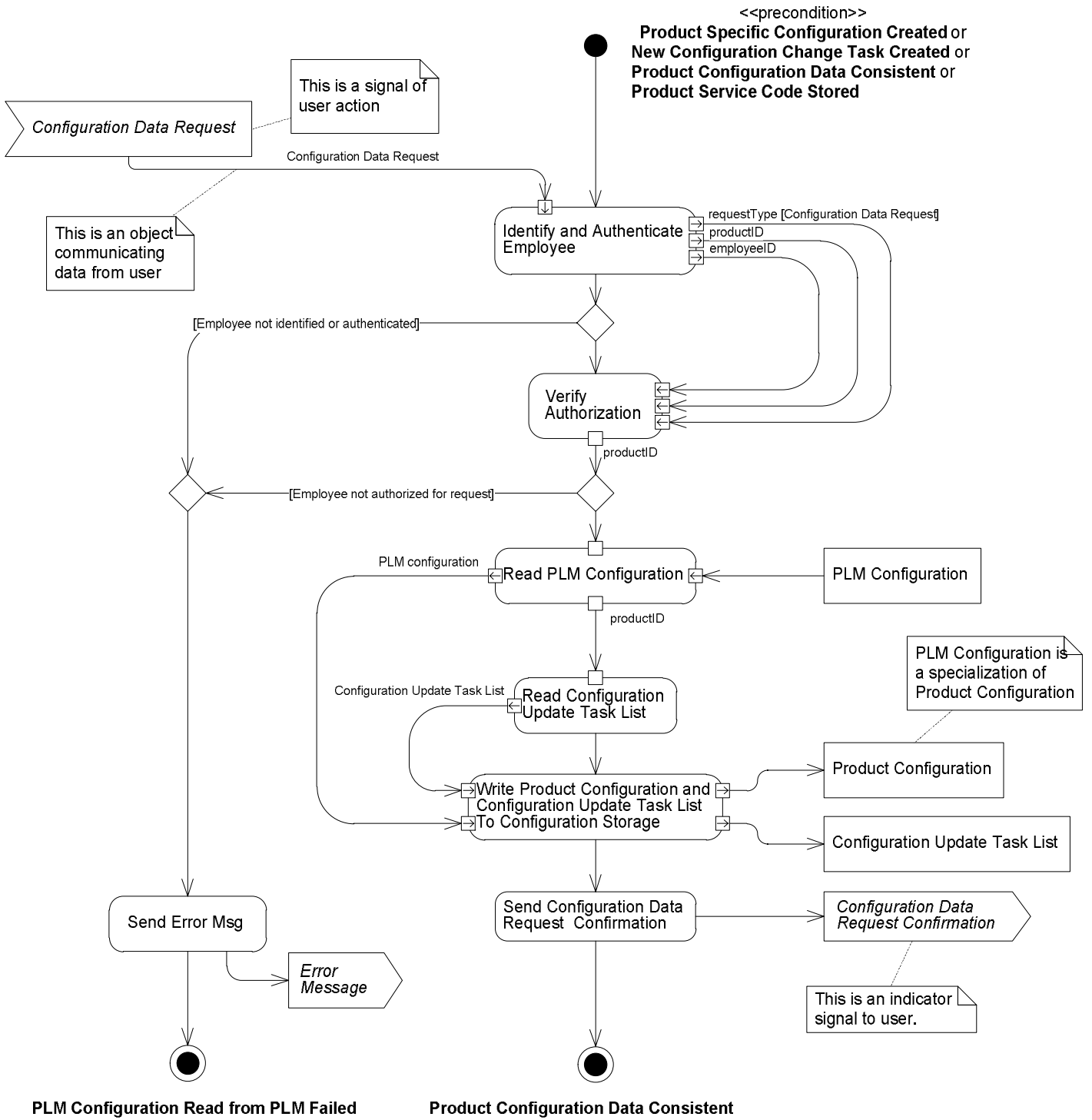
Product Configuration Not Updated

Product Configuration Updated

FILENAME	PROMONET_DCM_USECASEFLOWS_11.VSD	DRAWN BY	PARKKILA TOMMI, ISTO PEKKA	DATE	5/17/2012
		PAGE	6 OF 27	REVISED	5/14/2013

Print to A3 size for readability

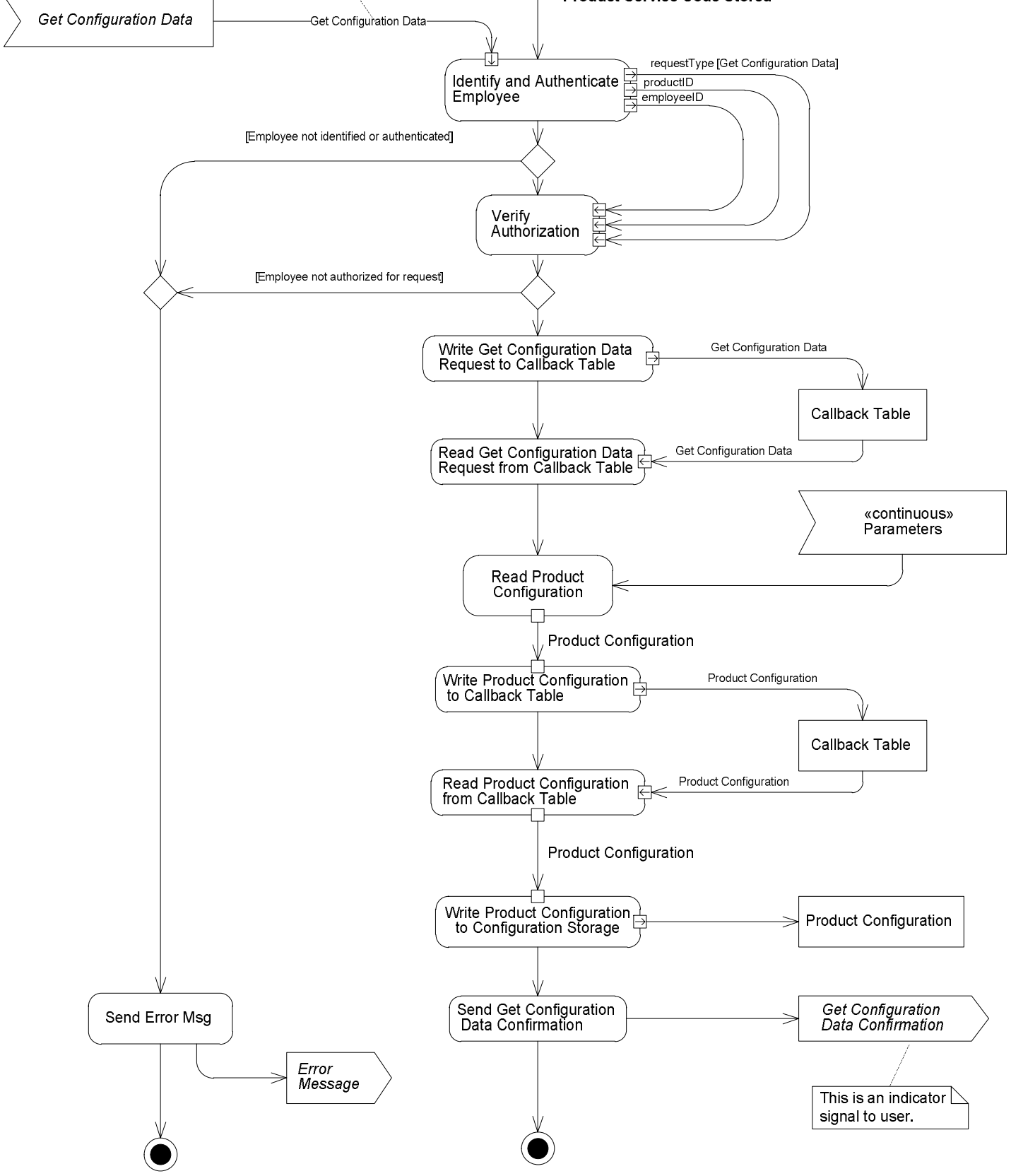




This is a signal of user action

This is an object communicating data from user

<<precondition>>  
**Product Specific Configuration Created or New Configuration Change Task Created or Product Configuration Data Consistent or Product Service Code Stored**

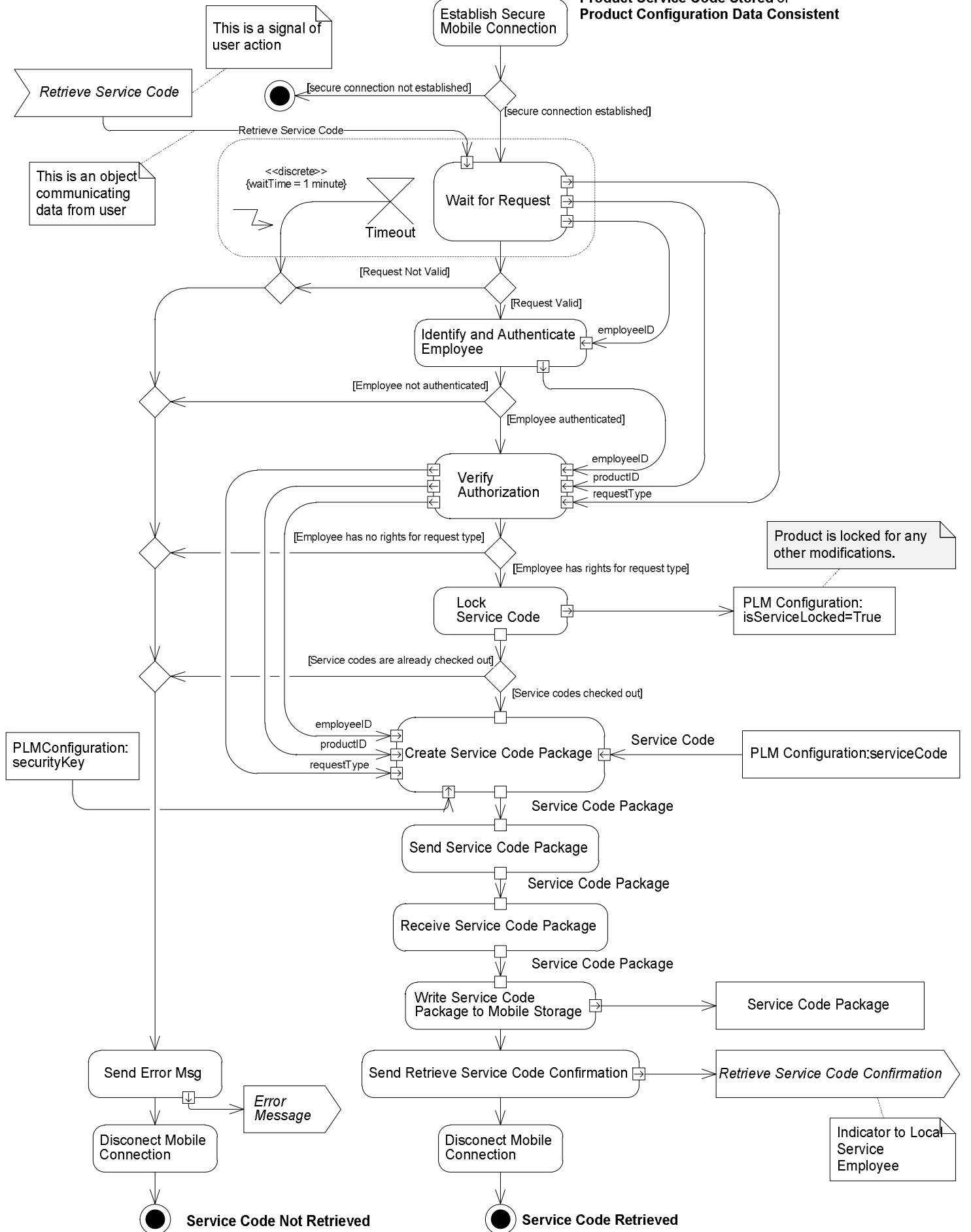


PLM Configuration Read from PLM Failed

Product Configuration Data Consistent

FILENAME PROMONET_DCM_USECASEFLOWS_11.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 5/17/2012
	PAGE 9 OF 27	REVISED 5/14/2013

Product Specific Configuration Created or  
New Configuration Change Task Created or  
Product Service Code Stored or  
Product Configuration Data Consistent



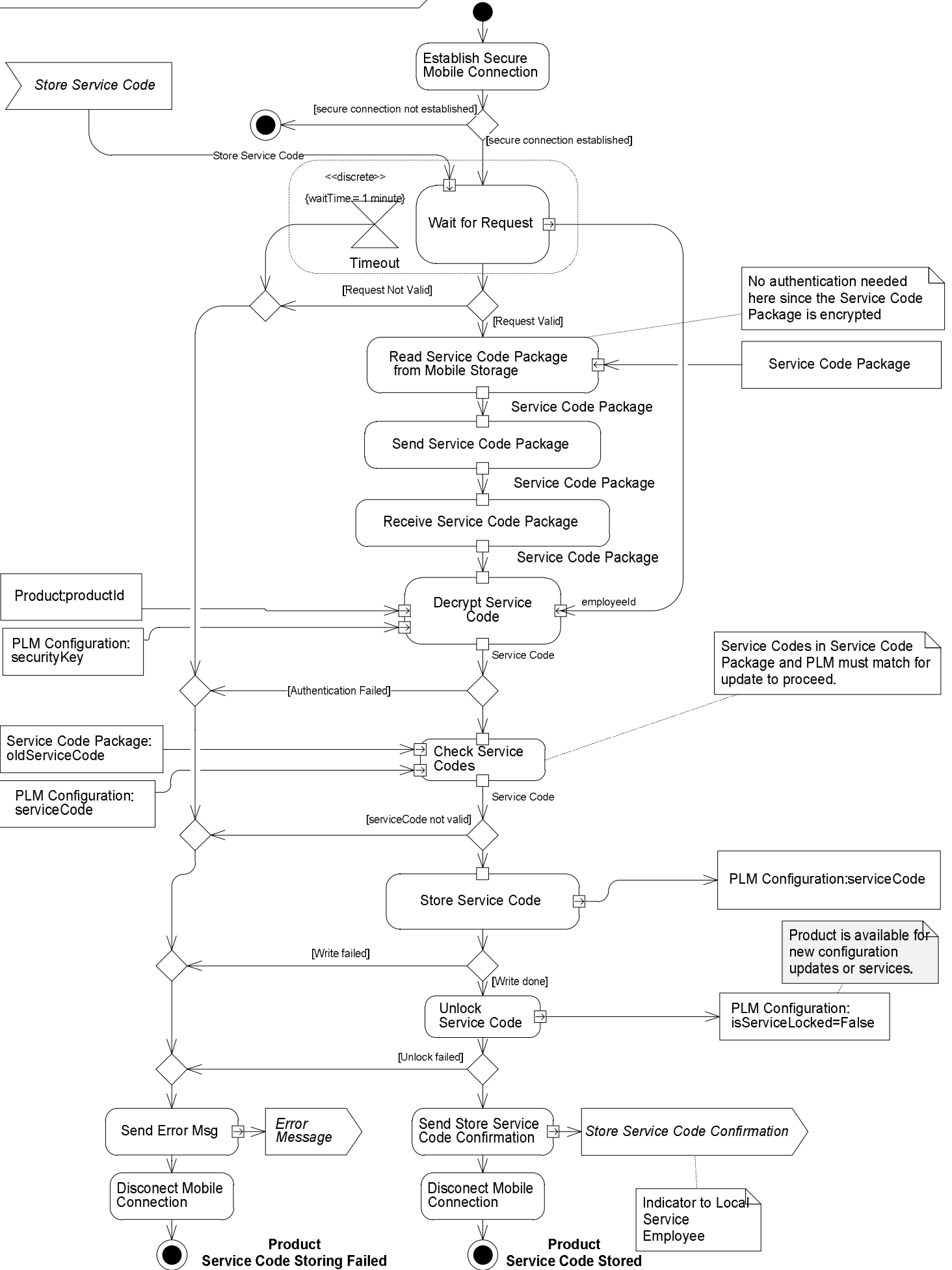
This is a signal of user action

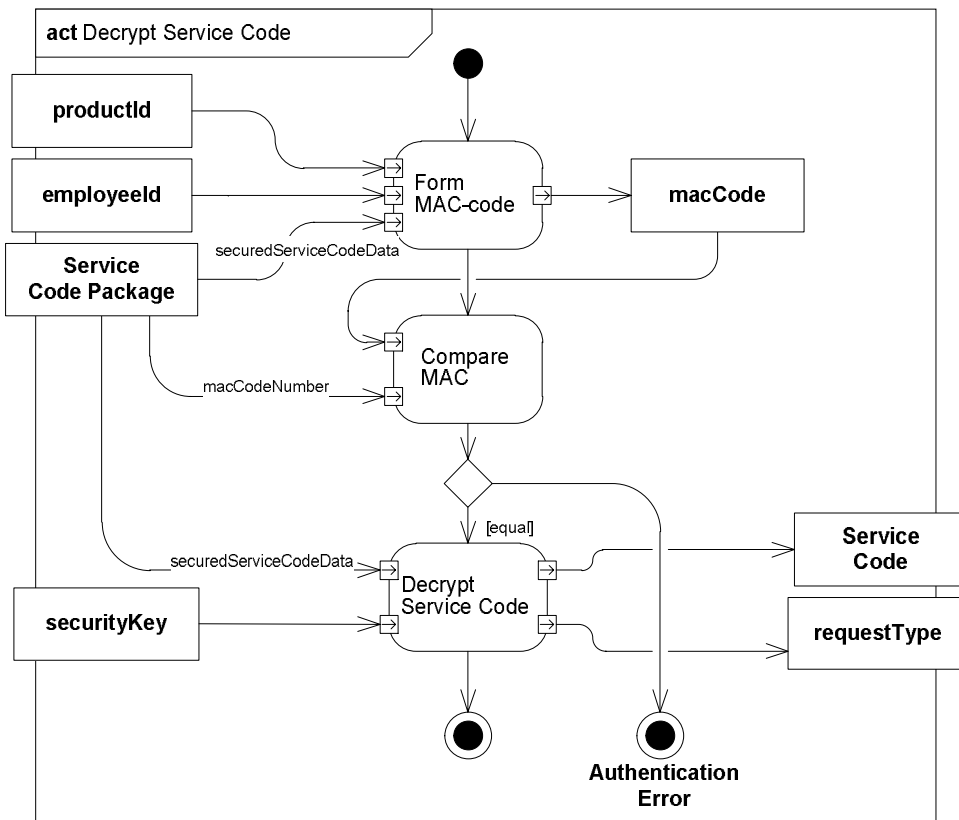
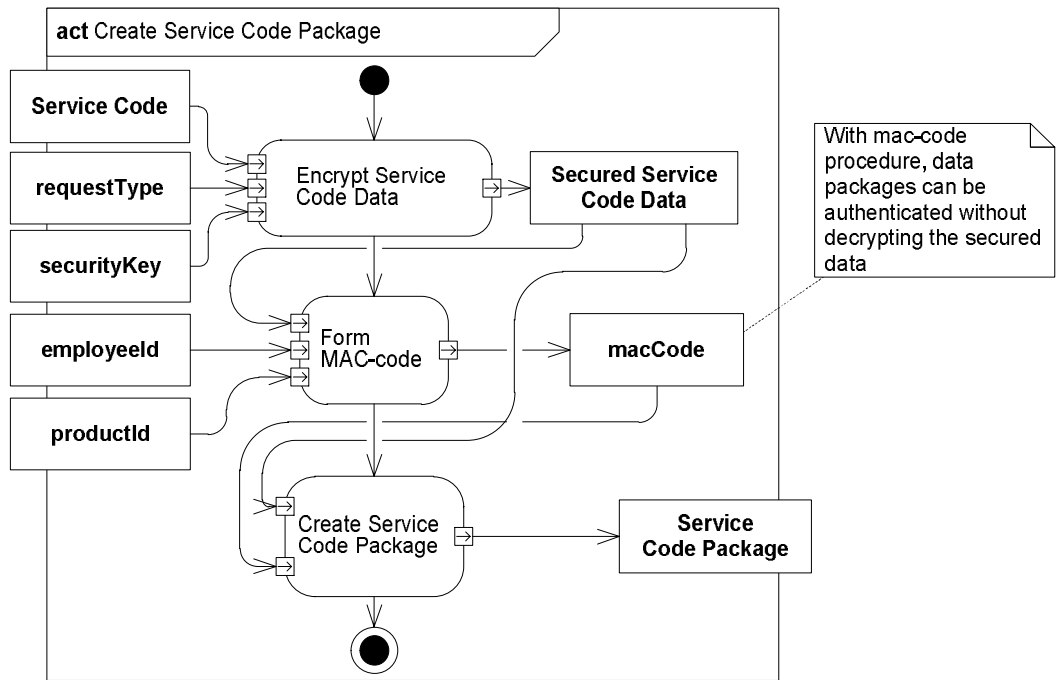
This is an object communicating data from user

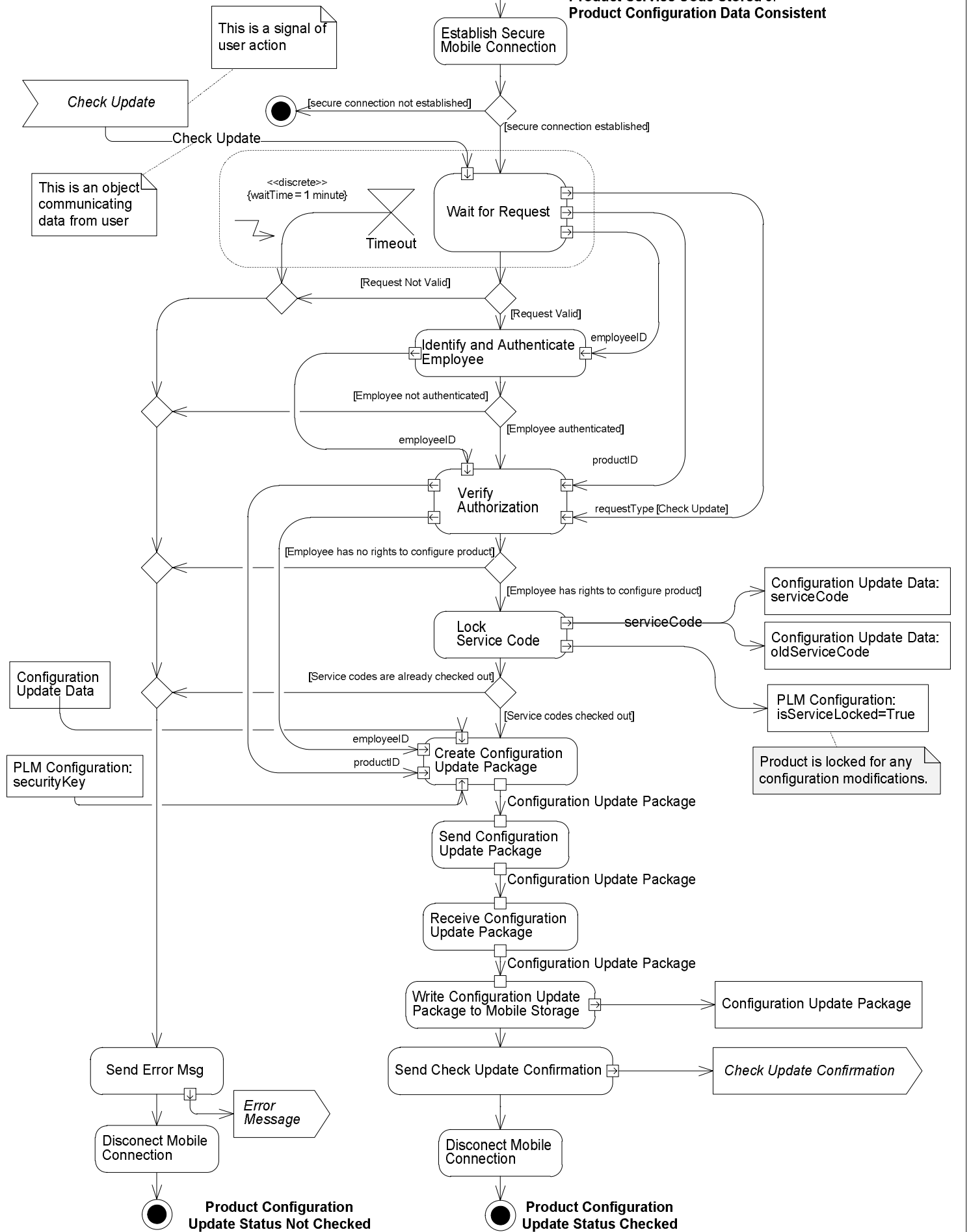
Product is locked for any other modifications.

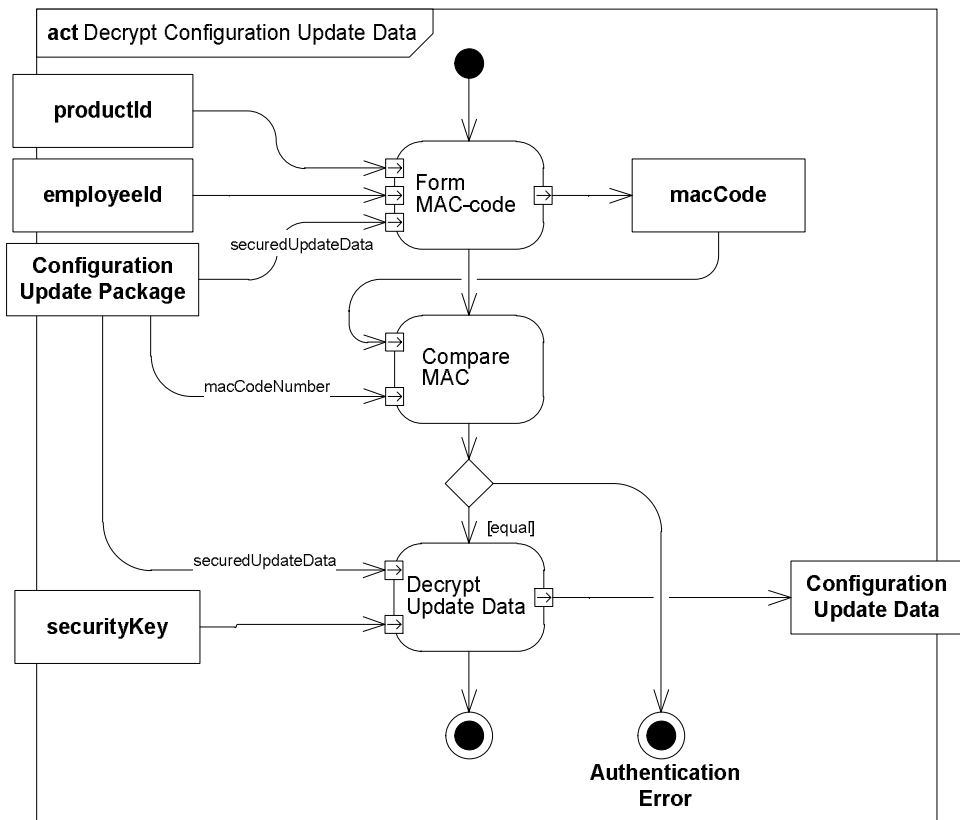
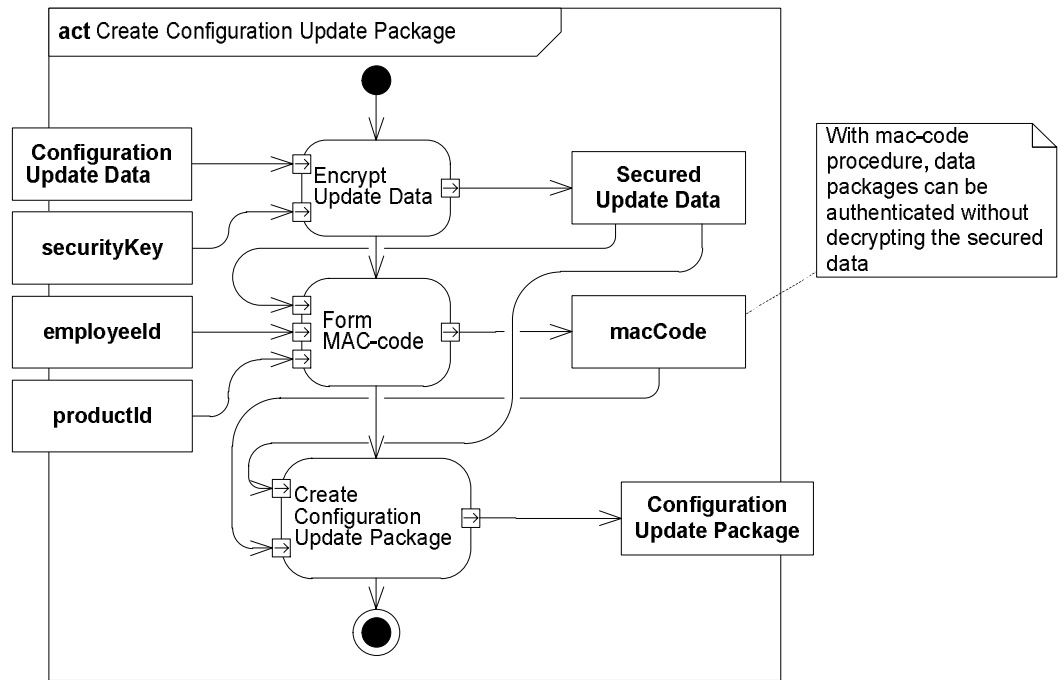
Indicator to Local Service Employee



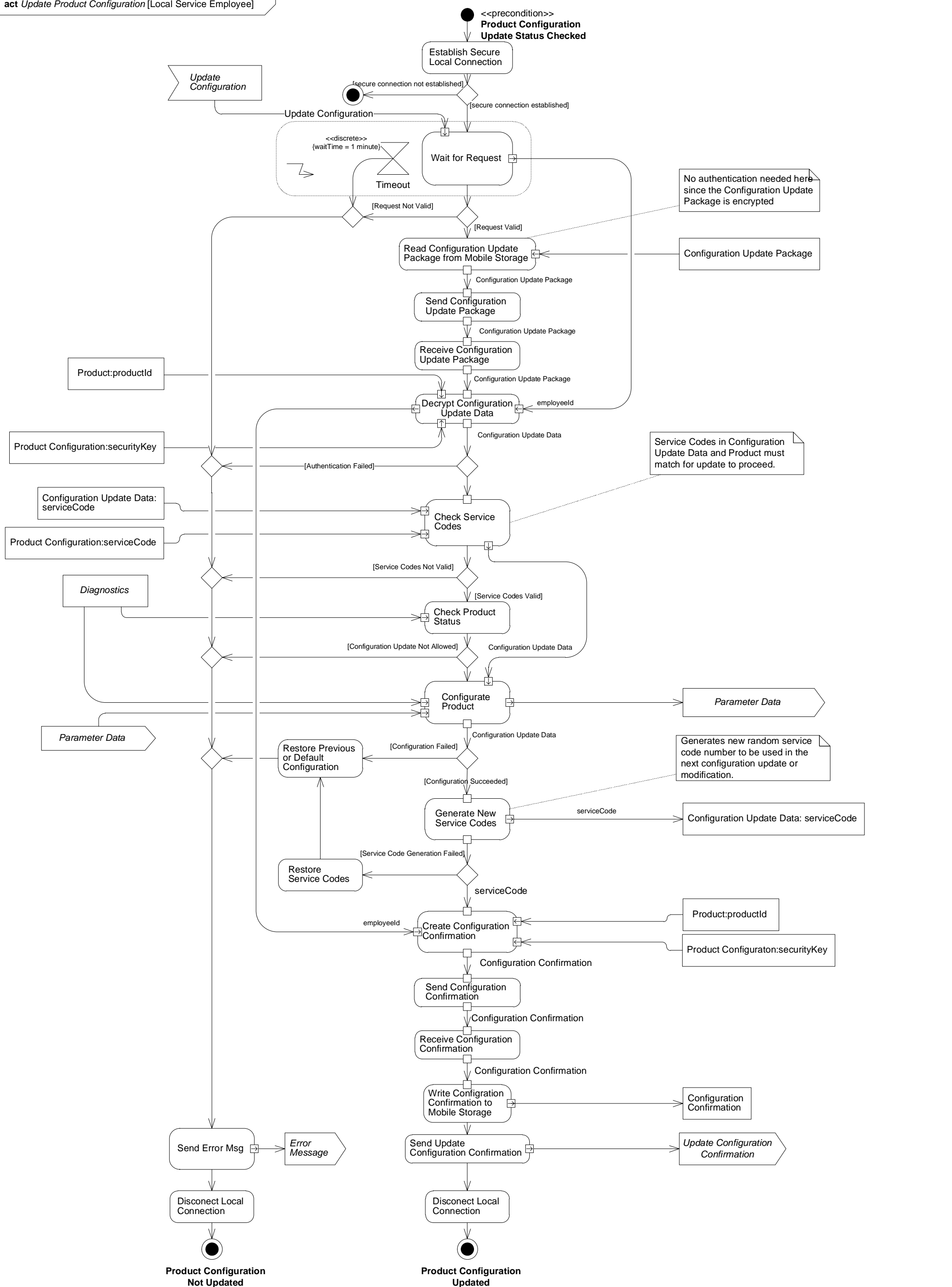








FILENAME PROMONET_DCM_USECASEFLOWS_11.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 5/17/2012
	PAGE 14 OF 27	REVISED 5/14/2013



No authentication needed here since the Configuration Update Package is encrypted

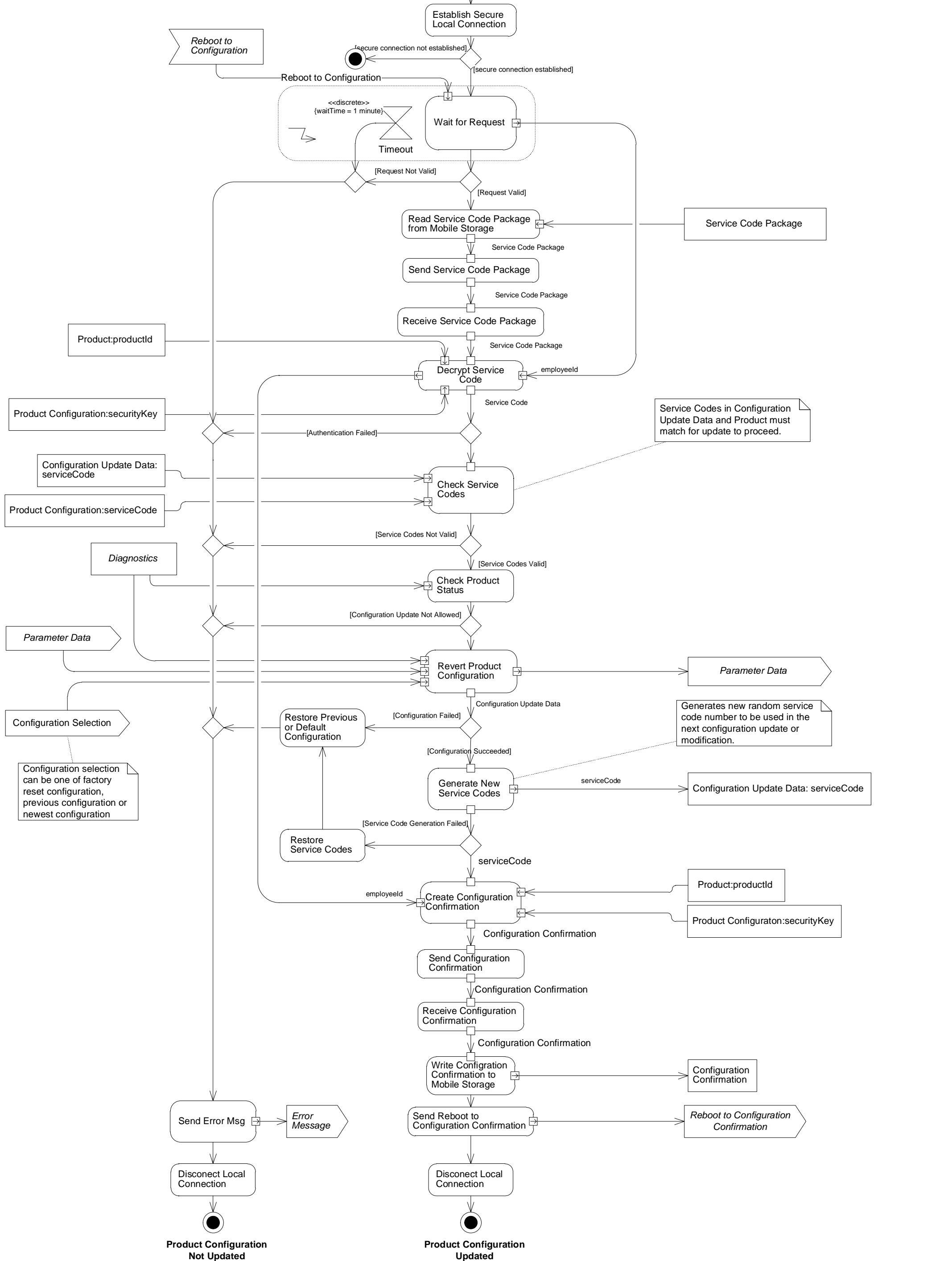
Configuration Update Package

Service Codes in Configuration Update Data and Product must match for update to proceed.

Generates new random service code number to be used in the next configuration update or modification.

Print to A3 size for readability

<<precondition>>  
Service Code Package Retrieved



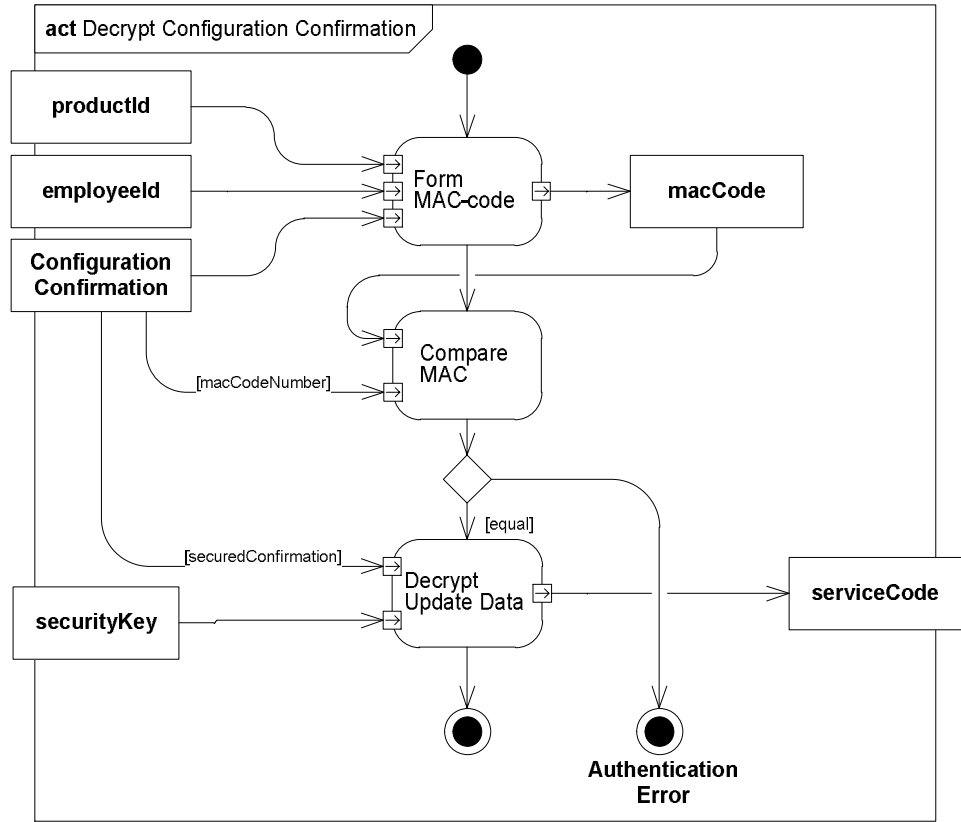
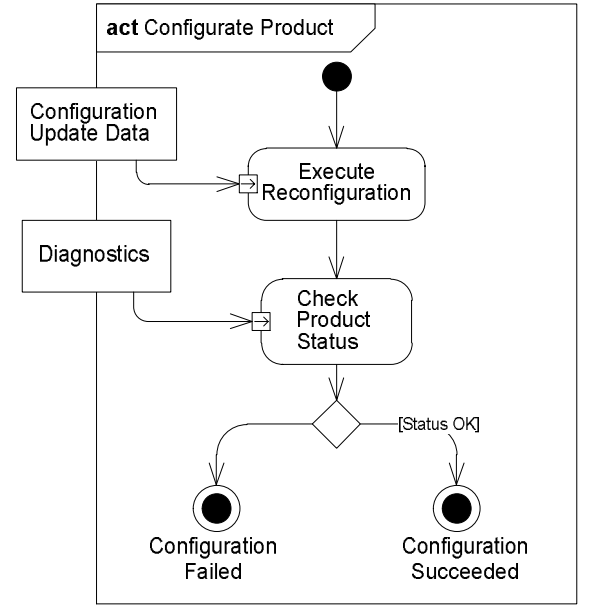
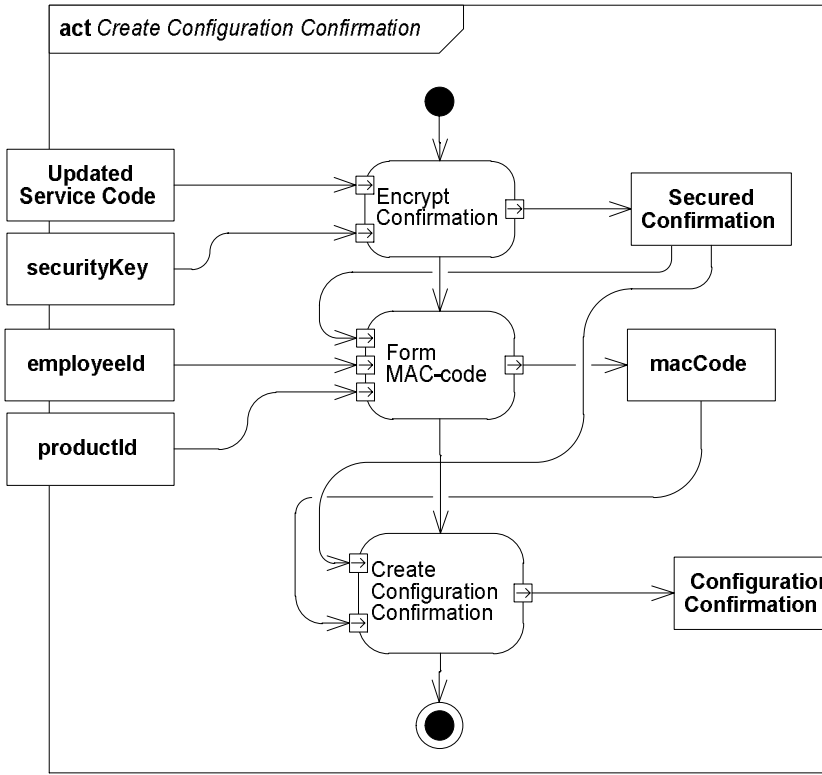
Service Codes in Configuration Update Data and Product must match for update to proceed.

Generates new random service code number to be used in the next configuration update or modification.

Configuration selection can be one of factory reset configuration, previous configuration or newest configuration

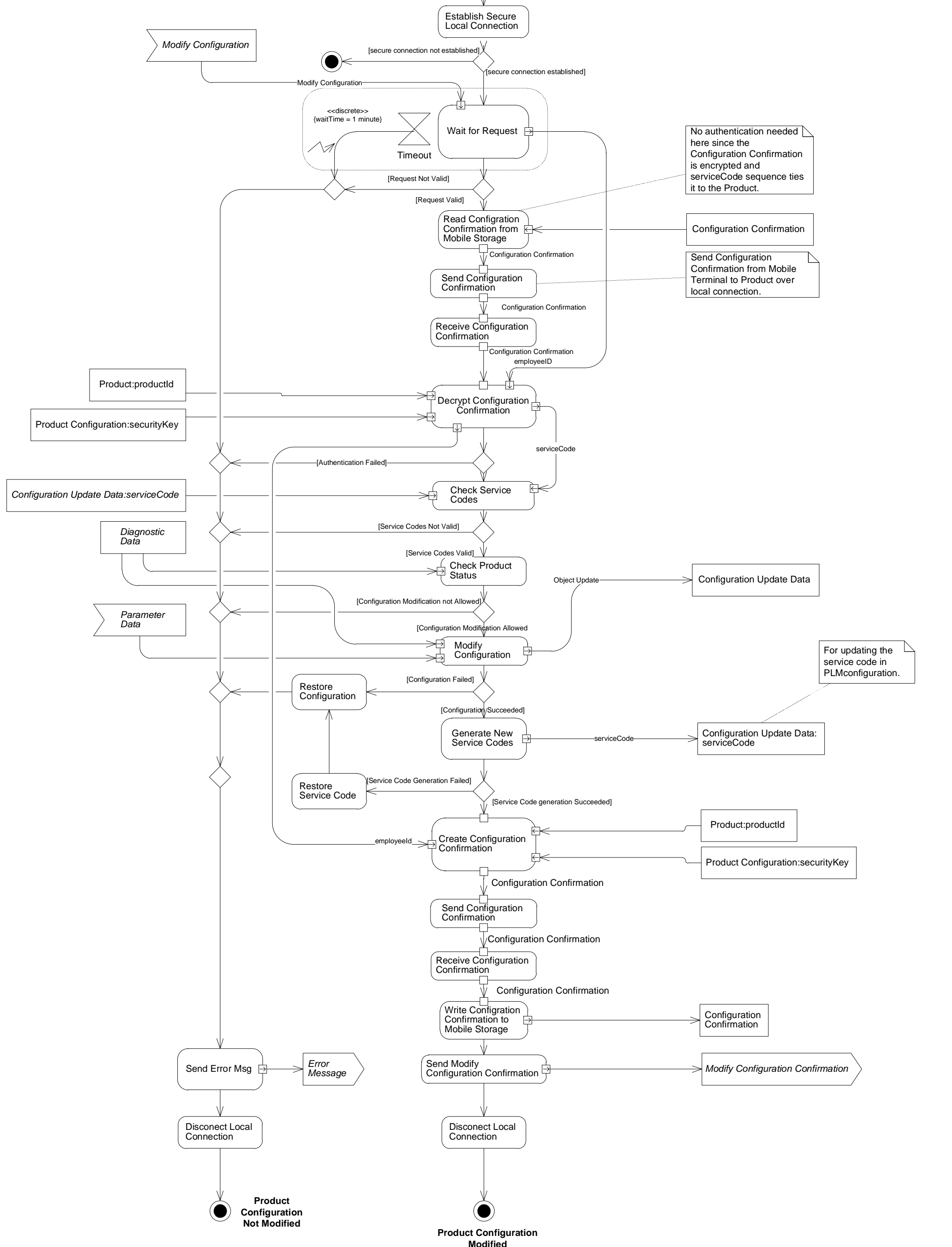
Print to A3 size for readability

FILENAME	DRAWN BY	DATE
PROMONET_DCM_USECASEFLOWS_11.VSD	PARKKILA TOMMI, ISTO PEKKA	5/17/2012
	PAGE	REVISED
	16 OF 27	5/14/2013



FILENAME PROMONET_DCM_USECASEFLOWS_11.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 5/17/2012
	PAGE 17 OF 27	REVISED 5/14/2013

<<precondition>>  
**Product Configuration Updated or  
 Product Configuration Modified or  
 Product Configuration Not Modified**



No authentication needed here since the Configuration Confirmation is encrypted and serviceCode sequence ties it to the Product.

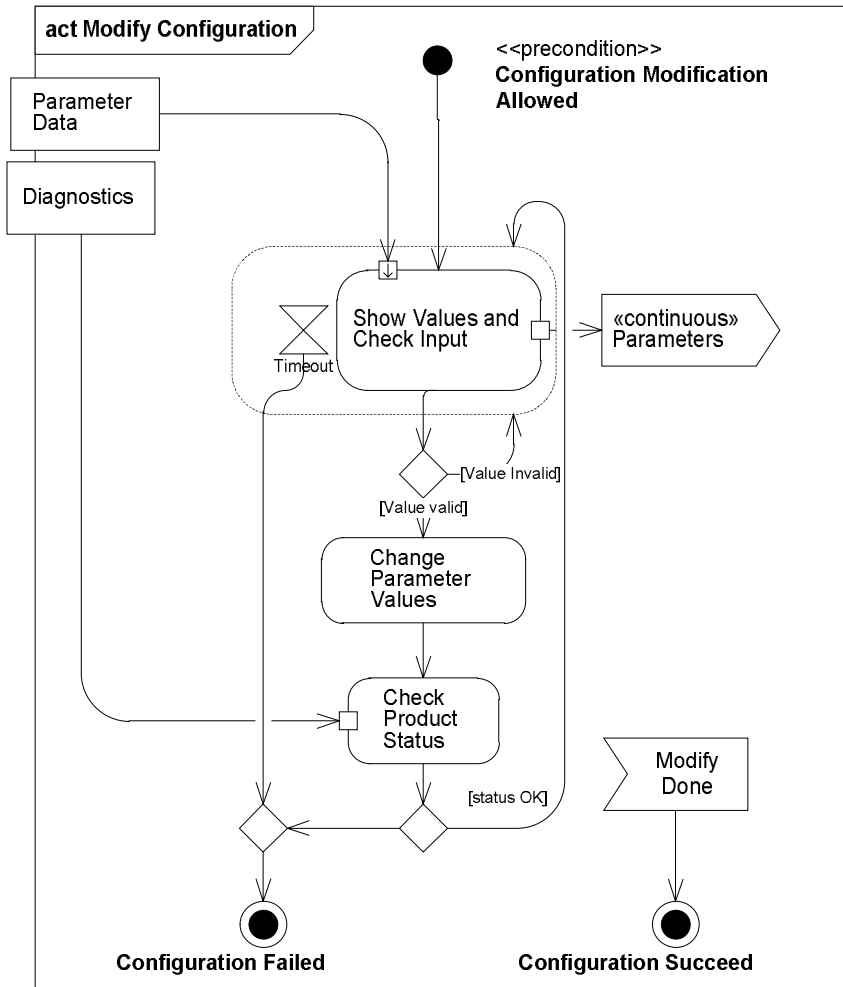
Configuration Confirmation  
 Send Configuration Confirmation from Mobile Terminal to Product over local connection.

For updating the service code in PLMconfiguration.

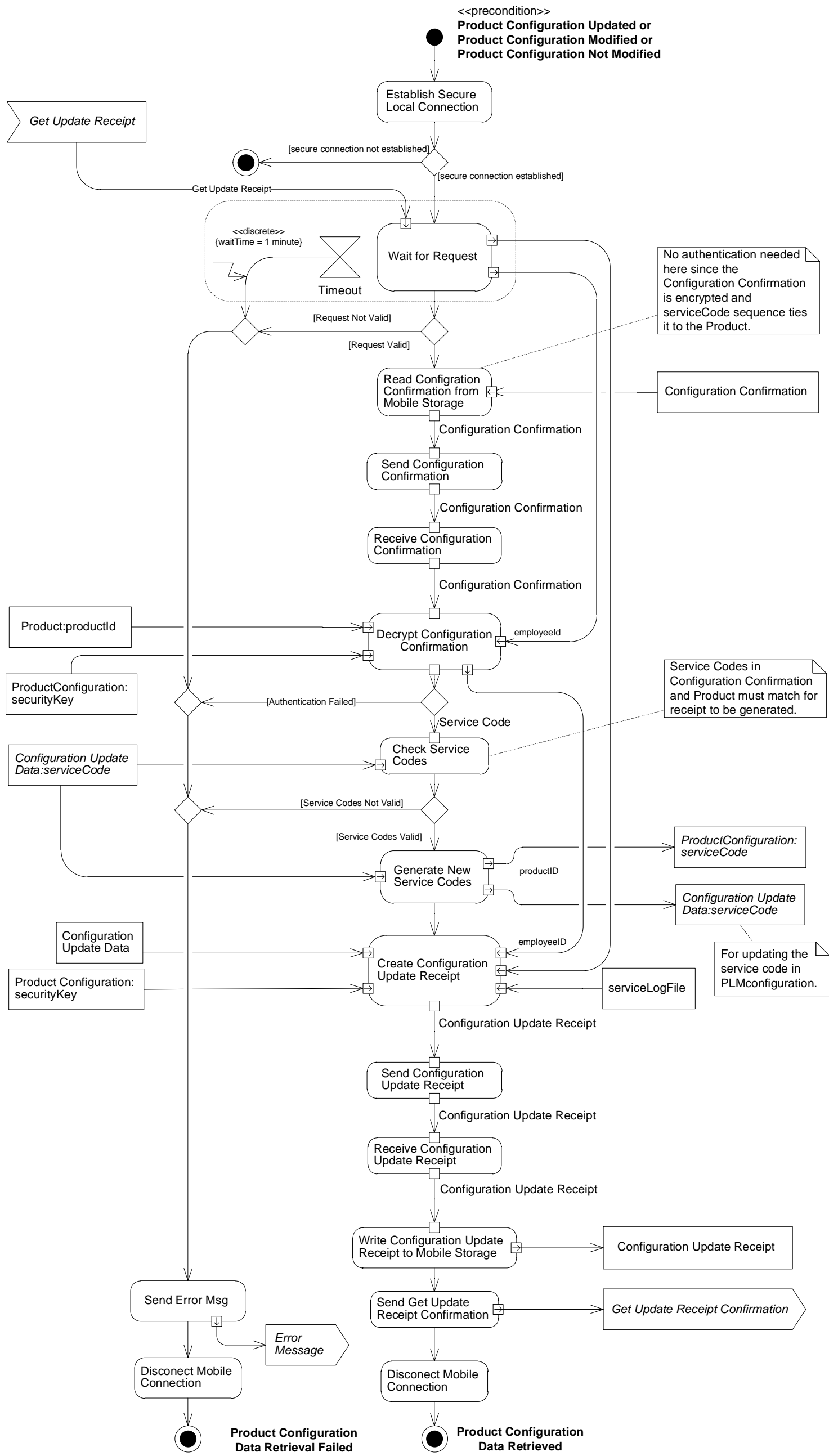
Print to A3 size for readability

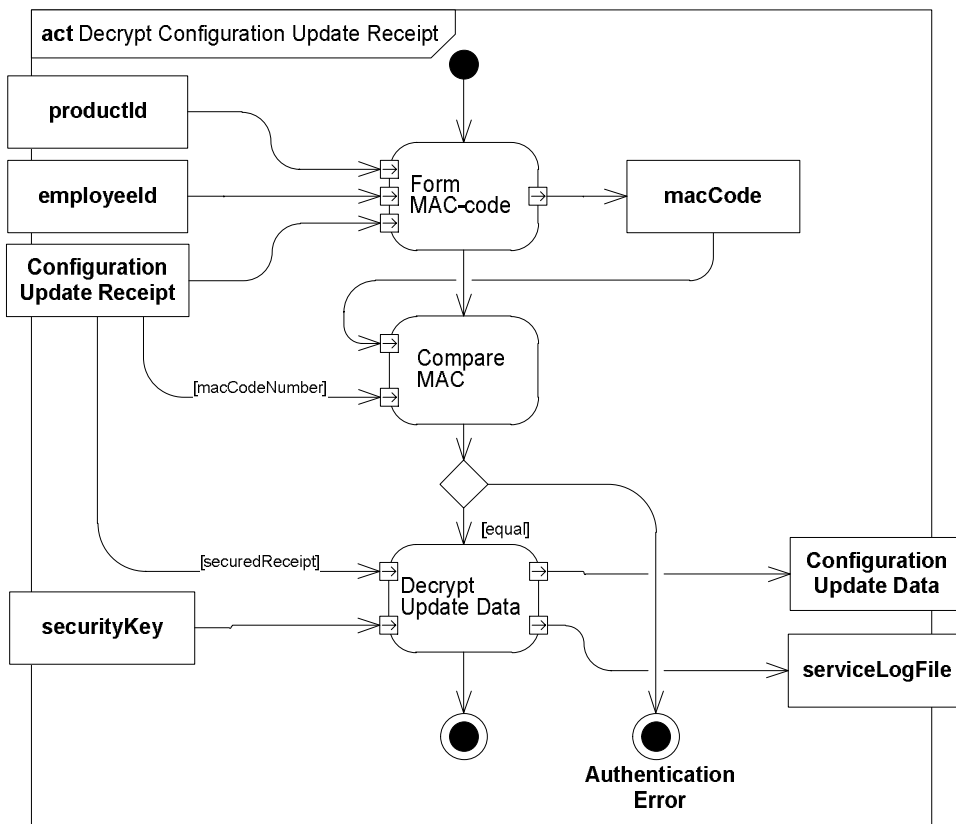
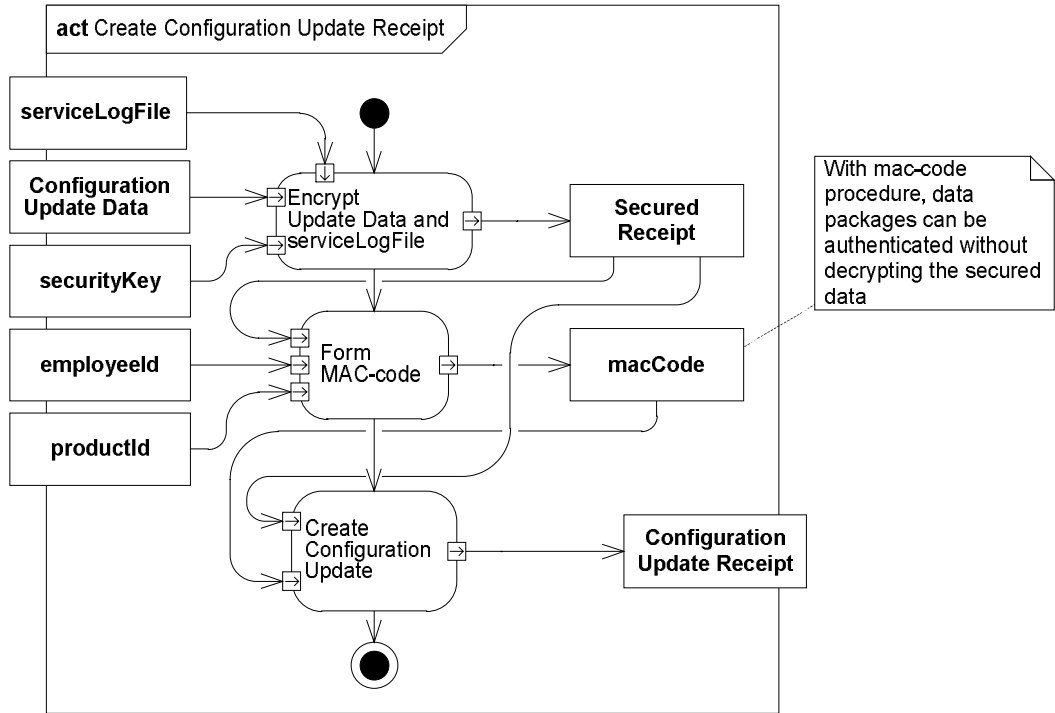
FILENAME	PROMONET_DCM_USECASEFLOWS_11.VSD	DRAWN BY	PARKKILA TOMMI, ISTO PEKKA	DATE	5/17/2012
		PAGE	18 OF 27	REVISED	5/14/2013



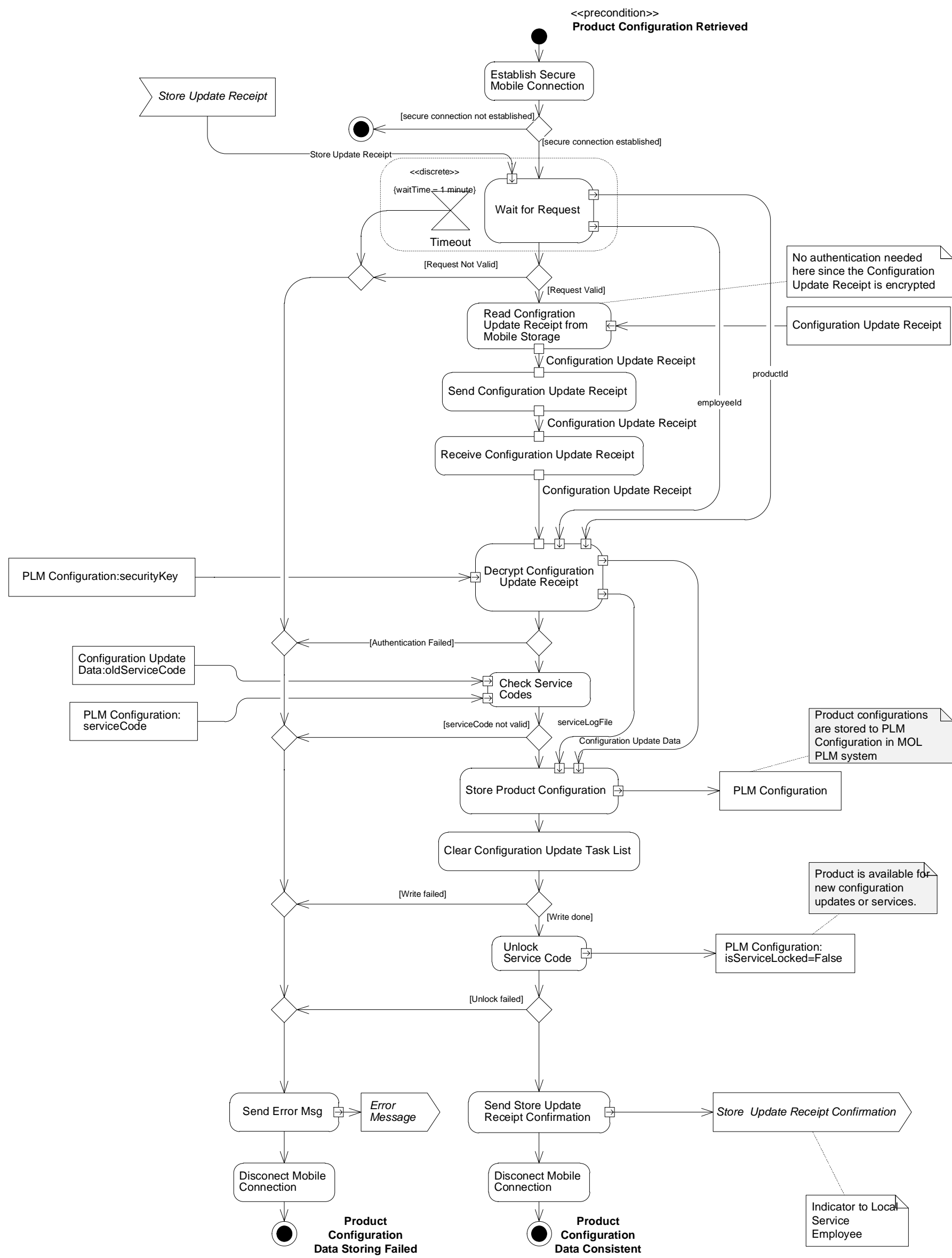


FILENAME PROMONET_DCM_USECASEFLOWS_11.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 5/17/2012
	PAGE 19 OF 27	REVISED 5/14/2013



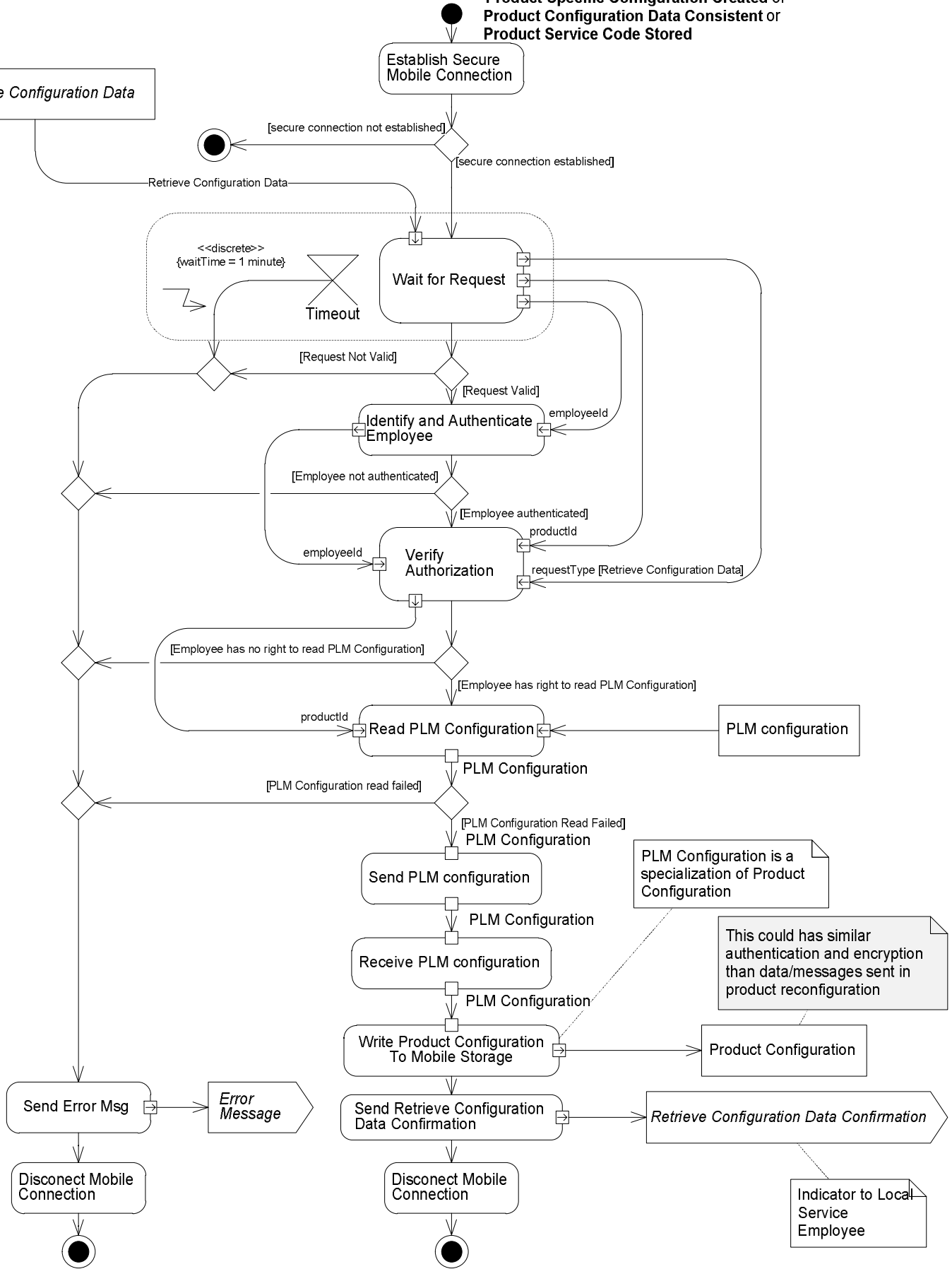


FILENAME PROMONET_DCM_USECASEFLOWS_11.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 5/17/2012
	PAGE 21 OF 27	REVISED 5/14/2013



<<precondition>>  
**Product Specific Configuration Created or  
 Product Configuration Data Consistent or  
 Product Service Code Stored**

Retrieve Configuration Data



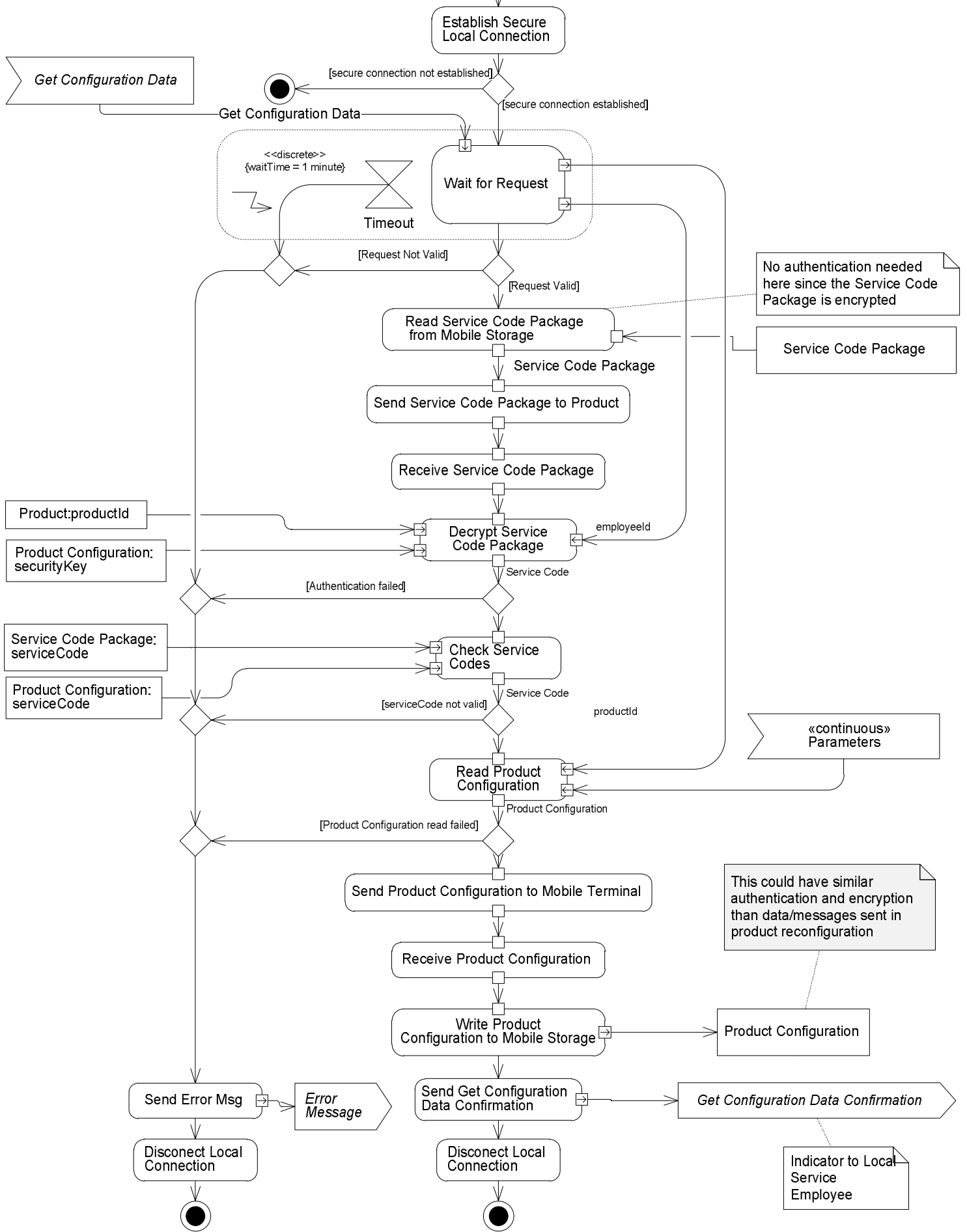
**PLM Configuration  
 Read from PLM Failed**

**Product Configuration Data Consistent**

FILENAME PROMONET_DCM_USECASEFLOWS_11.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 5/17/2012
	PAGE 23 OF 27	REVISED 5/14/2013

act Get Current Configuration Data From Product [Local Service Employee]

<<precondition>>  
Service Code Retrieved



No authentication needed here since the Service Code Package is encrypted

Service Code Package

«continuous» Parameters

This could have similar authentication and encryption than data/messages sent in product reconfiguration

Indicator to Local Service Employee

Product Configuration Data Read from Product Failed

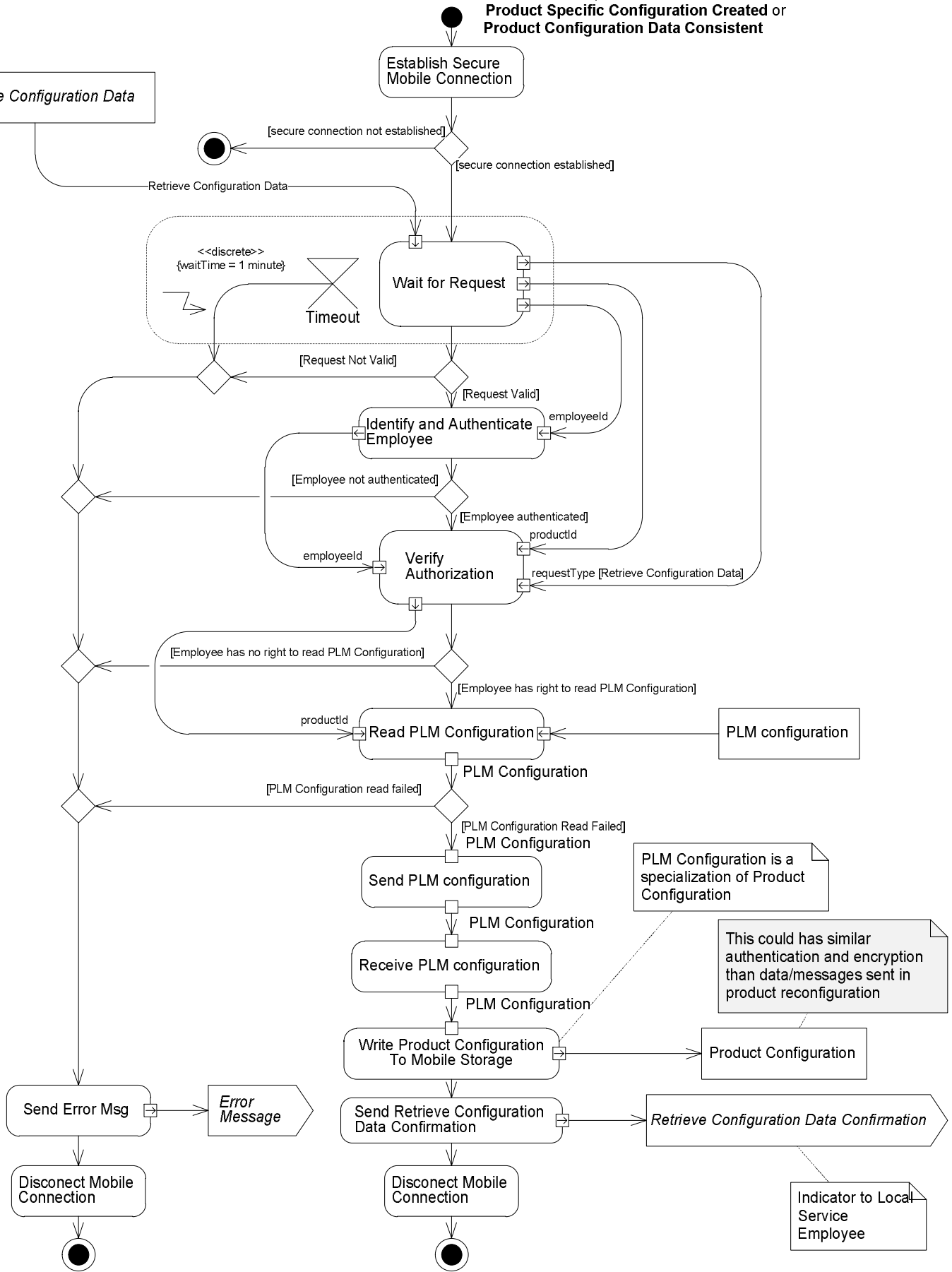
Product Configuration Data Consistent

FILENAME PROMONET_DCM_USECASEFLOWS_11.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 5/17/2012
	PAGE 24 OF 27	REVISED 5/14/2013

<<precondition>>

Product Specific Configuration Created or Product Configuration Data Consistent

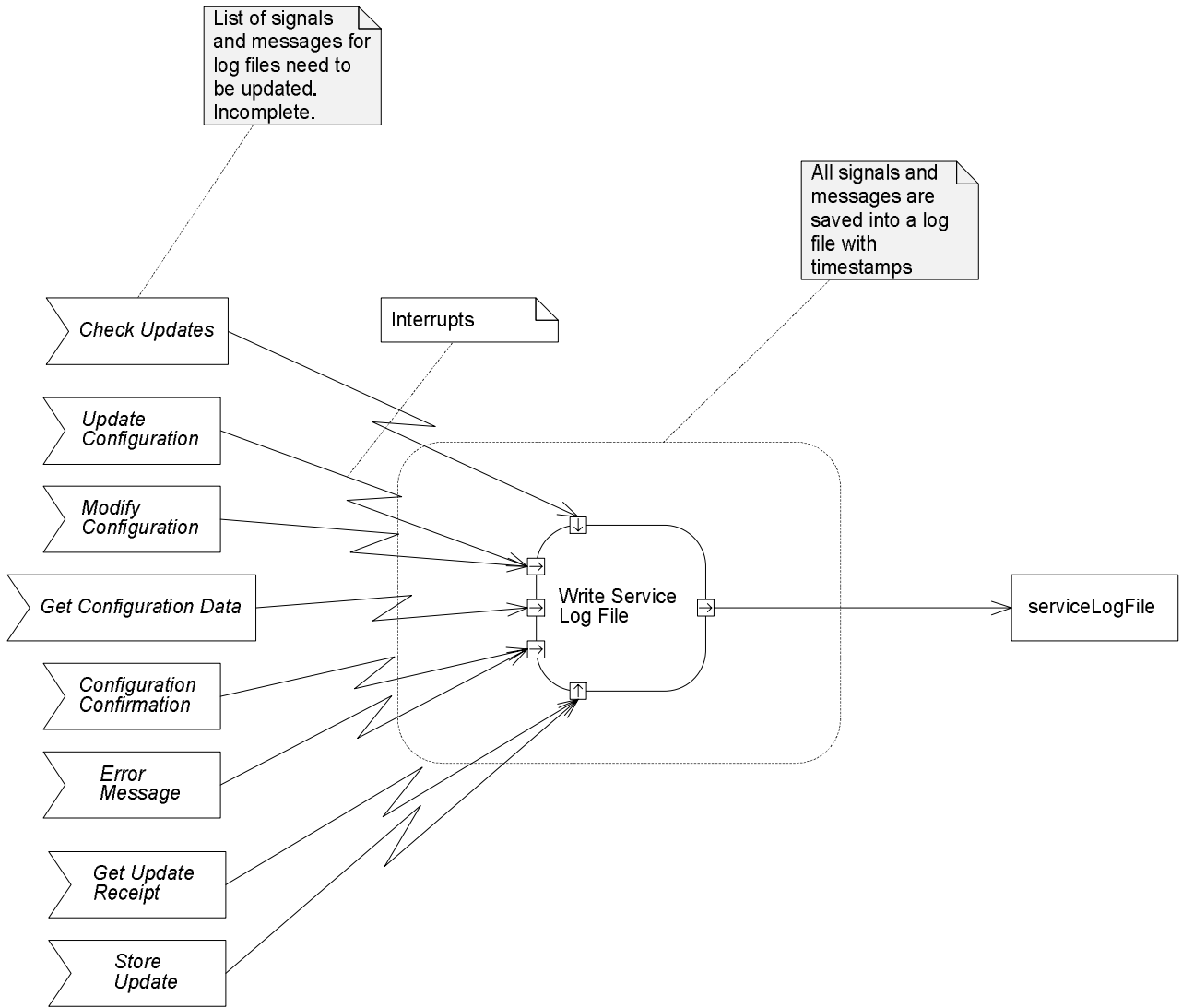
Retrieve Configuration Data



PLM Configuration Read from PLM Failed

Product Configuration Data Consistent

FILENAME PROMONET_DCM_USECASEFLOWS_11.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 5/17/2012
	PAGE 25 OF 27	REVISED 5/14/2013



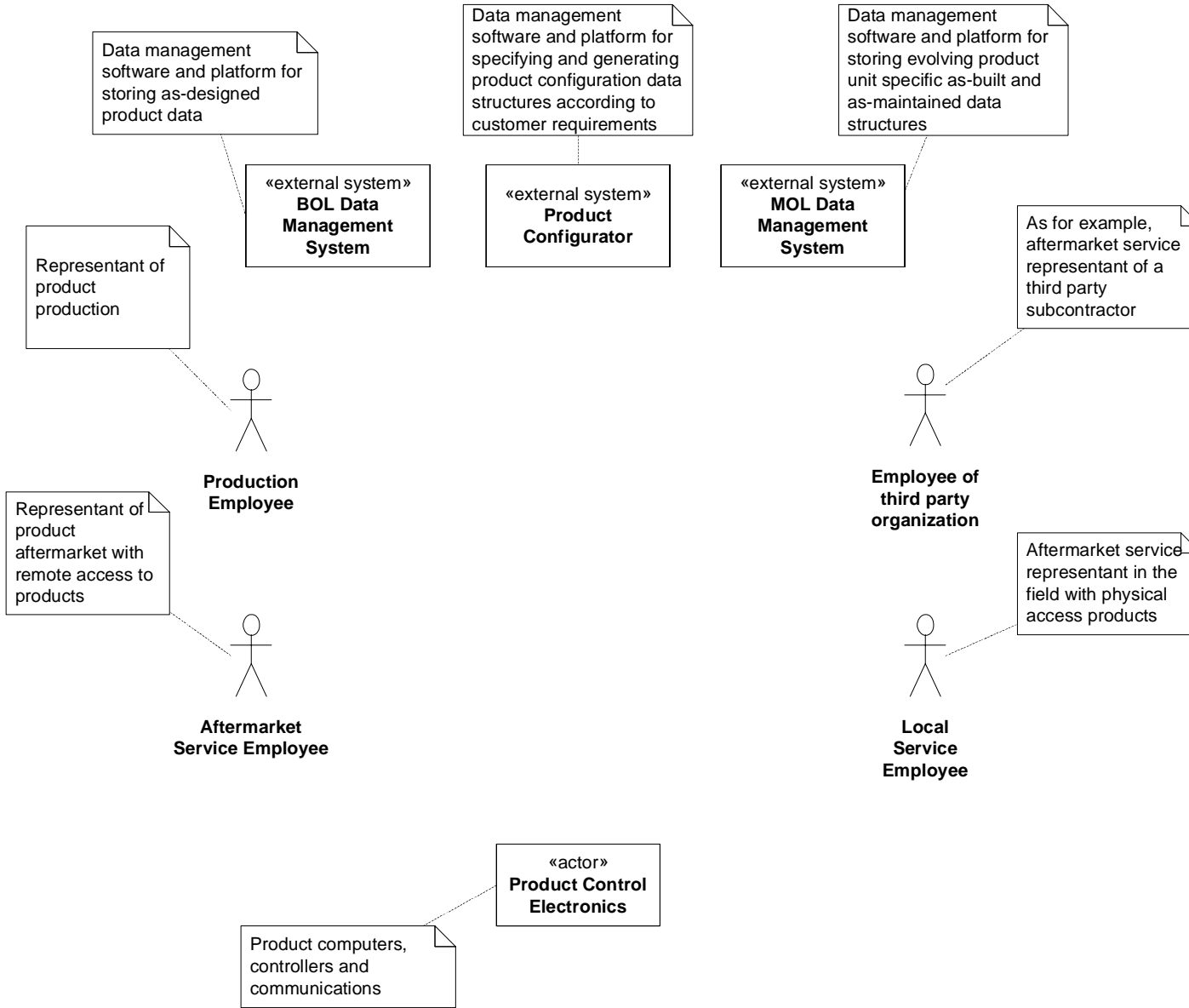


FILENAME PROMONET_DCM_USECASEFLOWS_11.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 5/17/2012
	PAGE 27 OF 27	REVISED 2/26/2013

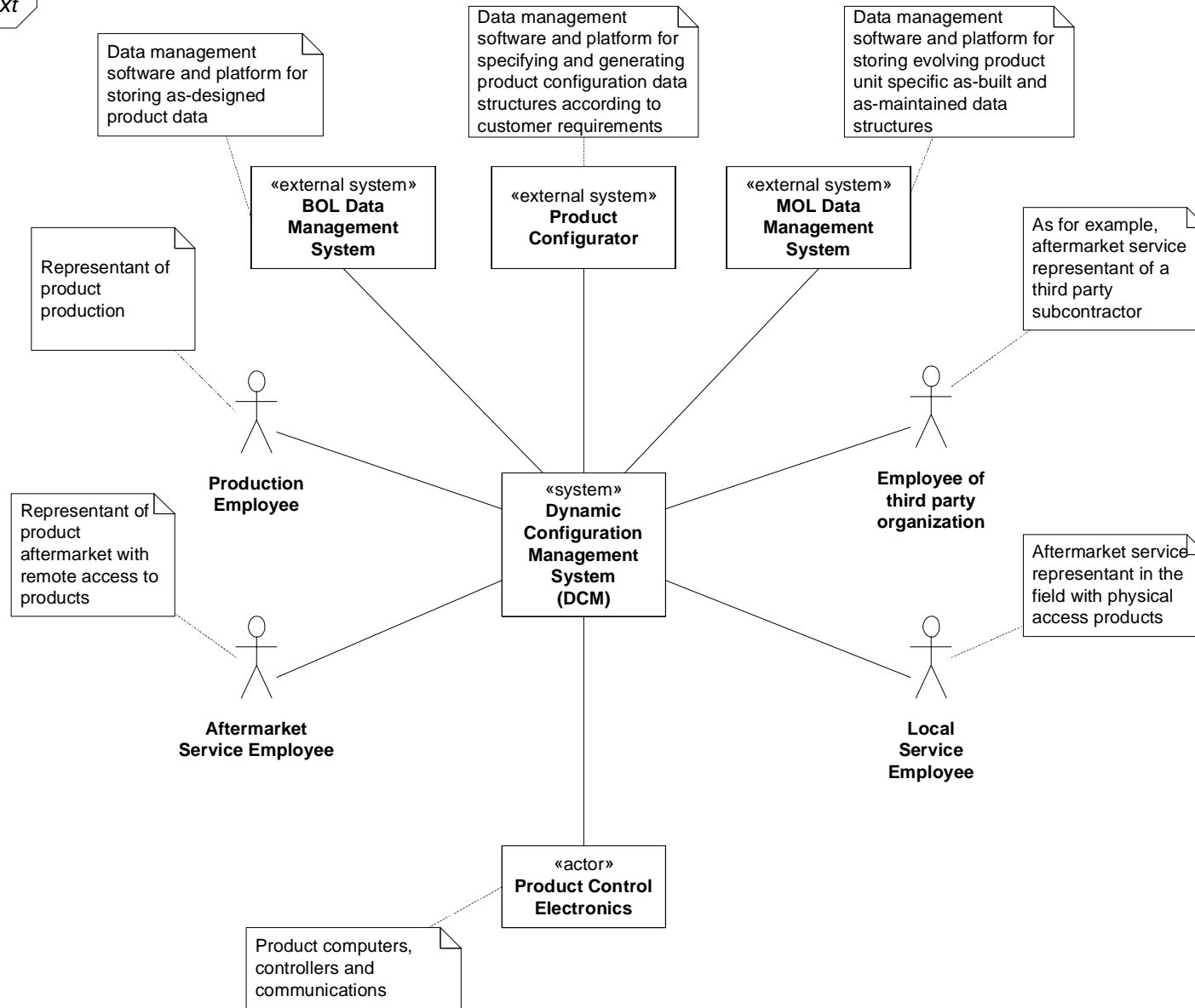
TITLE	ProMoNet DCM System Context		DESCRIPTION	System context diagram of ProMoNet Dynamic Configuration Management (DCM) system for reconfigurable networked industrial products		
FILENAME	PROMONET_DCM_SYSTEM_CONTEXT_05.VSD		DRAWN BY	PARKKILA TOMMI, ISTO PEKKA	DATE	4/25/2012
			PAGE	1 OF 8	REVISED	3/13/2013

REVISIONS			
REV.	DESCRIPTION	DATE	BY
0.1	First draft created	05/08/2012	TOP
0.2	Information flows edited, actor names changed, etc.	05/15/2012	TOP
0.4	Local Service part synchronized with, use cases (rev. 0.6), and flows (rev 0.4)	09/26/2012	TOP
0.5	Added Product Configurator and internal system structure.	03/13/2012	PEI

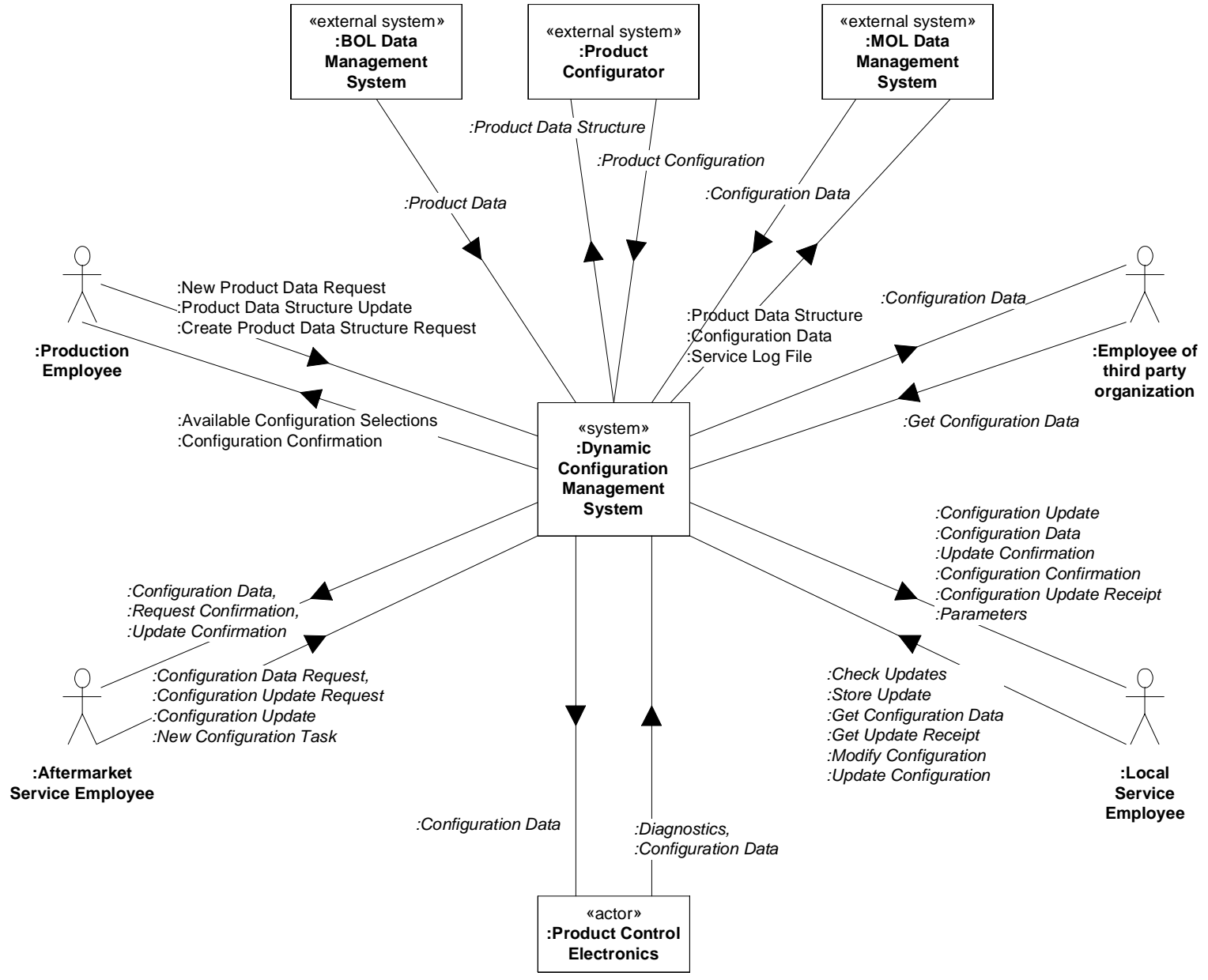
**bdd System Actors**



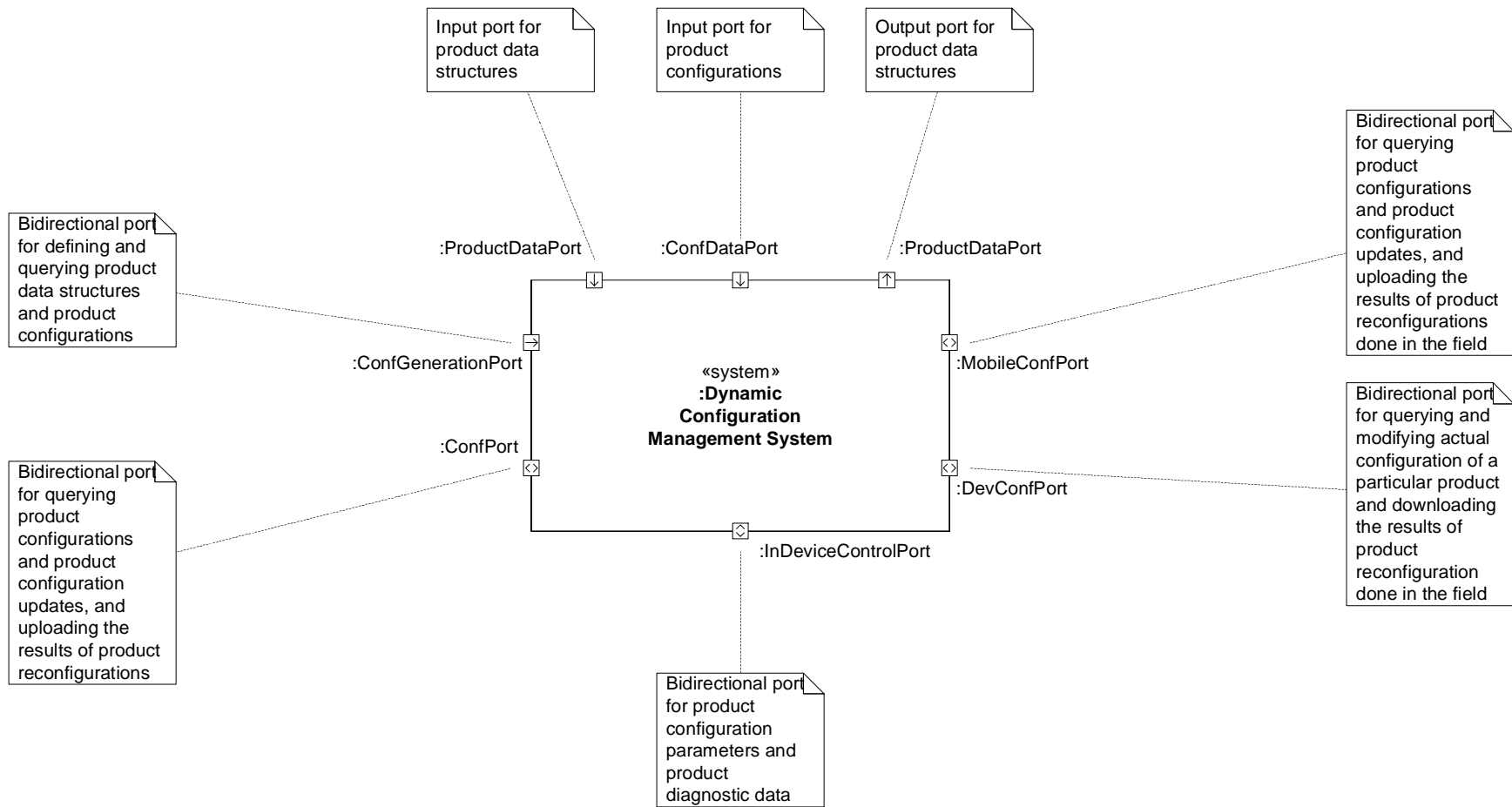
FILENAME PROMONET_DCM_SYSTEM_CONTEXT_05.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 4/25/2012
	PAGE 2 OF 8	REVISED 3/13/2013



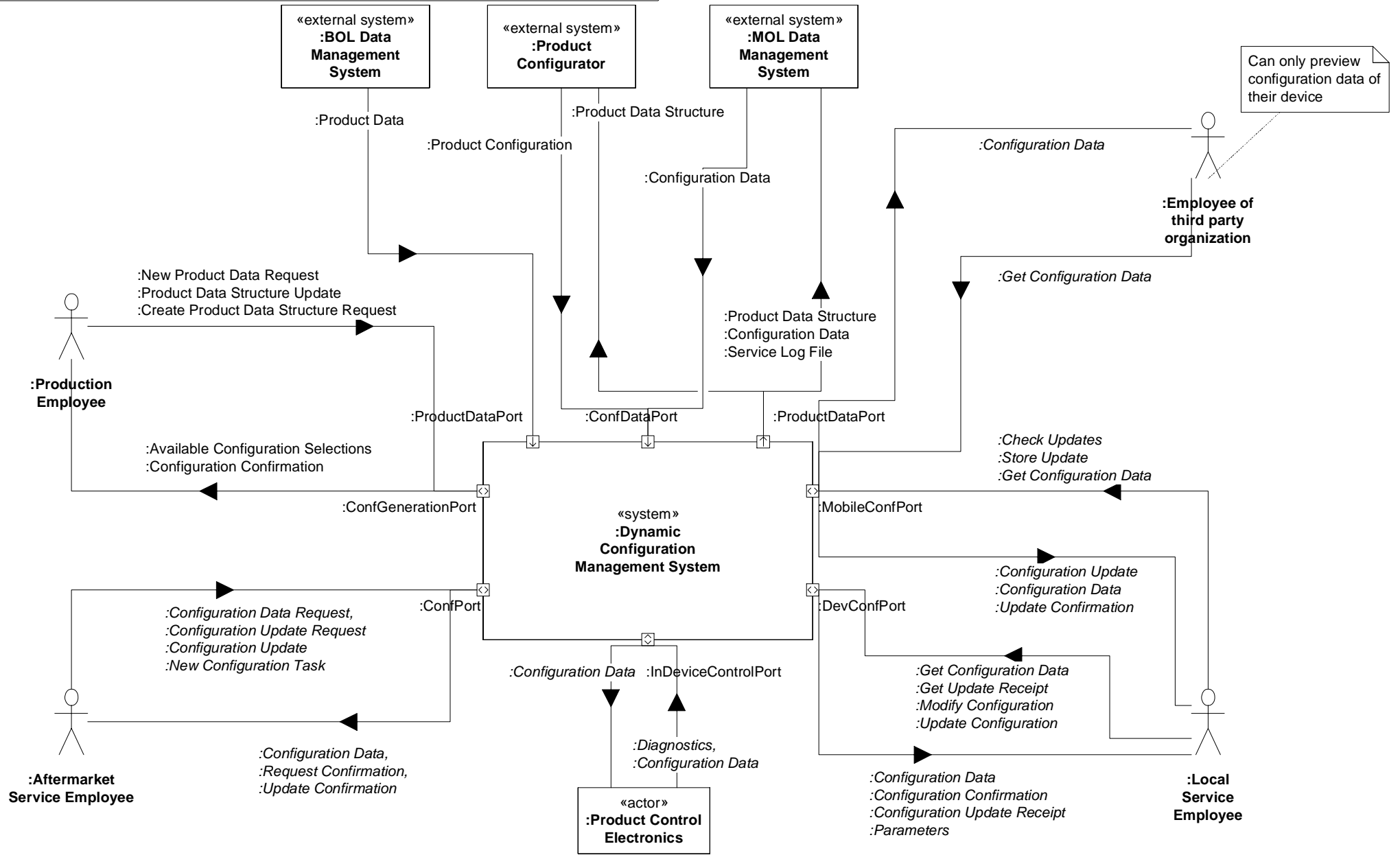
FILENAME PROMONET_DCM_SYSTEM_CONTEXT_05.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 4/25/2012
	PAGE 3 OF 8	REVISED 3/13/2013



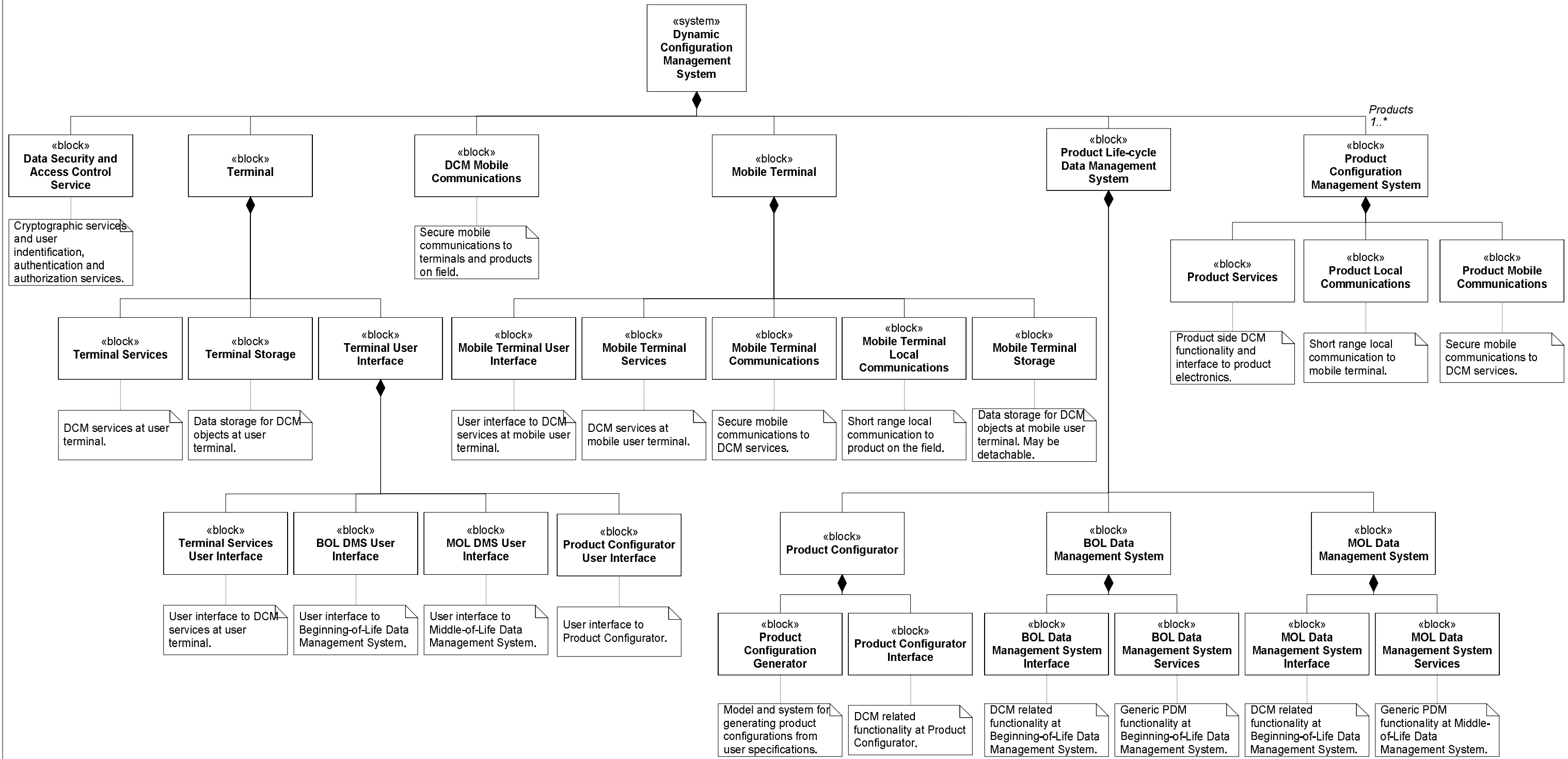
FILENAME	DRAWN BY	DATE
PROMONET_DCM_SYSTEM_CONTEXT_05.VSD	PARKKILA TOMMI, ISTO PEKKA	4/25/2012
	PAGE	REVISED
	4 OF 8	3/13/2013



FILENAME PROMONET_DCM_SYSTEM_CONTEXT_05.VSD	DRAWN BY PARKKILA TOMMI, ISTO PEKKA	DATE 4/25/2012
	PAGE 5 OF 8	REVISED 3/13/2013



FILENAME	DRAWN BY	DATE
PROMONET_DCM_SYSTEM_CONTEXT_05.VSD	PARKKILA TOMMI, ISTO PEKKA	4/25/2012
	PAGE	REVISED
	6 OF 8	3/13/2013



Print in A3 size for readability. Note that this sheet consist of two layers one for the decomposition diagram and one for comments which can be made separately visible and printable from the appropriate Visio menu.

FILENAME	PROMONET_DCM_SYSTEM_CONTEXT_05.VSD	DRAWN BY	PARKKILA TOMMI, ISTO PEKKA	DATE	4/25/2012
		PAGE	7 OF 8	REVISED	3/13/2013



