IoT@Work

WP 2 – COMMUNICATION NETWORKS
D2.1 – IOT ADDRESSING SCHEMES APPLIED TO MANUFACTURING

<table>
<thead>
<tr>
<th>Reference:</th>
<th>IoT@Work/WP2/D2.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category:</td>
<td>Report</td>
</tr>
<tr>
<td>Deliverable Responsible:</td>
<td>SIEMENS</td>
</tr>
<tr>
<td>Author(s):</td>
<td>A. M. Houyou, H.-P. Huth, S. Mechs G. Völksen, H.–J. Hof, J. Gessner (Siemens) Christos Kloukinas, Igor Siveroni (City) H. Trsek (inIT)</td>
</tr>
<tr>
<td>Due Date:</td>
<td>30/11/2010</td>
</tr>
<tr>
<td>Deliverable Date:</td>
<td>11/10/2010 Revised on: 20/12/2011</td>
</tr>
<tr>
<td>Project Start Date:</td>
<td>01/06/2010</td>
</tr>
<tr>
<td>Project Duration:</td>
<td>36 Months</td>
</tr>
<tr>
<td>Status:</td>
<td>Revised</td>
</tr>
<tr>
<td>Availability:</td>
<td>Public</td>
</tr>
</tbody>
</table>
Executive Summary

The deliverable D2.1 is the first step towards the definition of a bootstrapping mechanism that implements the IoT@Work proposed Plug and Work approach. For this purpose the deliverable looks first at the enabling technologies found in the Internet and more specifically the latest developments towards the Internet of Things (IoT). The deliverable will enlist the defined IoT addressing and naming schemes that could be integrated in IoT@Work. Ideas related to service and application semantics are defined. The requirements of Plug&Work are investigated with the help of the scenarios provided in Deliverable D1.1 (State of the Art and Initial Requirements). The naming and addressing requirements of the three scenarios are detailed in this document as well.

The deliverable is structured in the following way. After the Introduction in Section 1, Section 2 starts by making an analysis of the naming and addressing problematic in the Internet. A focus is made on developments made to enable the Internet of Things. Industrial and factory automation adopted protocols for naming and addressing are explained in more details. Section 3 presents service discovery mechanisms existing in higher layers and that are often integrated in middleware approaches. The goal of the section is to analyze the role of middleware for embedded and distributed systems in the setup of self-organizing system. The use of directories for both names, service instances, and resources are also an important element of architecting the IoT, therefore Section 4 looks into the evolution of directory systems. The need for a secure bootstrap and the existing security mechanisms are detailed in Section 5. In Section 6, configuration tools are introduced to explain the current practice in configuring names and addresses in factory automation systems. Section 7 analyses the needs of auto-configuration, locator and identifier separation problems and indirection needs identified in the IoT@Work scenarios.

Section 8 provides a list of trend analysis and current developments of state of the art that could affect both the bootstrapping architecture and tooling and or implementation details of proposed functionalities. The conclusion of the deliverable together with the next steps and that will use this deliverable are summarized in Section 9.
## Document History

### Version History

<table>
<thead>
<tr>
<th>Version</th>
<th>Status</th>
<th>Date</th>
<th>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>Draft</td>
<td>August 2010</td>
<td>A. Houyou (Siemens)</td>
</tr>
<tr>
<td>0.2</td>
<td>Draft</td>
<td>Sept. 2010</td>
<td>A. Houyou (Siemens)</td>
</tr>
<tr>
<td>0.3</td>
<td>Draft</td>
<td>Sept. 2010</td>
<td>H. P. Huth, S. Mechs, G. Voelksen, (Siemens), I. Siveroni (City), H. Trsek (inIT)</td>
</tr>
<tr>
<td>0.4</td>
<td>Draft</td>
<td>Oct 2010</td>
<td>S. Mechs, G. Voelksen, (Siemens)</td>
</tr>
<tr>
<td>0.5</td>
<td>Draft</td>
<td>Nov 2010</td>
<td>H.-J. Hof (Siemens)</td>
</tr>
<tr>
<td>0.6</td>
<td>Draft</td>
<td>Nov 2010</td>
<td>I. Siveroni (City), H. Trsek (inIT)</td>
</tr>
<tr>
<td>0.7</td>
<td>Draft</td>
<td>Nov 2010</td>
<td>A. Houyou, H. P. Huth, S. Mechs, H.-J. Hof (Siemens), I. Siveroni (City), J. Jasperneite, H. Trsek (inIT)</td>
</tr>
<tr>
<td>0.8</td>
<td>Draft</td>
<td>Nov 2010</td>
<td>A. Houyou, S. Mechs (Siemens), I. Siveroni (City), H. Trsek (inIT)</td>
</tr>
<tr>
<td>0.9</td>
<td>Under review</td>
<td>Nov 2010</td>
<td>A. Houyou, H. P. Huth, S. Mechs, H.-J. Hof (Siemens), I. Siveroni (City), J. Jasperneite, H. Trsek (inIT)</td>
</tr>
<tr>
<td>1.0</td>
<td>Final</td>
<td>Nov 2010</td>
<td>A. Houyou (Siemens)</td>
</tr>
<tr>
<td>1.1</td>
<td>Revised</td>
<td>Aug. 2011</td>
<td>A. Houyou (Siemens)</td>
</tr>
<tr>
<td>1.2</td>
<td>Revised</td>
<td>Oct. 2011</td>
<td>A. Houyou (Siemens)</td>
</tr>
<tr>
<td>1.3</td>
<td>Revised</td>
<td>Nov 2011</td>
<td>H. Trsek (inIT), G. Völksen, J. Gessner (Siemens)</td>
</tr>
<tr>
<td>1.4</td>
<td>Revised</td>
<td>Dec 2011</td>
<td>A. Houyou (Siemens), C. Kloukinas (CITY)</td>
</tr>
<tr>
<td>1.5</td>
<td>Revised</td>
<td>Dec 2011</td>
<td>A. Houyou (Siemens)</td>
</tr>
</tbody>
</table>

### Summary of Changes

<table>
<thead>
<tr>
<th>Version</th>
<th>Section(s)</th>
<th>Synopsis of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>Initial ToC</td>
<td>Document structure</td>
</tr>
<tr>
<td>0.2</td>
<td>Initial ToC</td>
<td>Revised after Telco</td>
</tr>
<tr>
<td>0.3</td>
<td>Final ToC</td>
<td>Added bullet points detailing content</td>
</tr>
<tr>
<td>0.4</td>
<td>Section 3, 4, 6</td>
<td>Added initial contributions from Siemens to sections 3, 4, and 6</td>
</tr>
<tr>
<td>0.5</td>
<td>Section 5</td>
<td>Added Siemens contributions</td>
</tr>
<tr>
<td>0.6</td>
<td>All Sections</td>
<td>Added contributions to all sections</td>
</tr>
<tr>
<td>0.7</td>
<td>All Sections</td>
<td>Added missing contributions</td>
</tr>
<tr>
<td>0.8</td>
<td>All Sections</td>
<td>Refinements and cross-checking</td>
</tr>
<tr>
<td>0.9</td>
<td>All Sections</td>
<td>Refinements following final reviews</td>
</tr>
<tr>
<td>-----</td>
<td>--------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>1.0</td>
<td>Document</td>
<td>Finalized Conclusion and final improvements</td>
</tr>
<tr>
<td>1.1</td>
<td>Document</td>
<td>Review of document in light of the reviewers’ comments</td>
</tr>
<tr>
<td>1.2</td>
<td>All Sections</td>
<td>Identifying editorial tasks, and modifying Section 2</td>
</tr>
<tr>
<td>1.3</td>
<td>All Sections</td>
<td>Adding a conclusion statement to each respective Section on how the State of the Art is used in IoT@Work. A new conclusion (i.e. Section 9) is written.</td>
</tr>
<tr>
<td>1.4</td>
<td>All Sections</td>
<td>Reviewing of new version.</td>
</tr>
<tr>
<td>1.5</td>
<td>Document</td>
<td>Last editing of final version and provision of Changes Summary.</td>
</tr>
</tbody>
</table>
# Contents

1 Introduction ............................................................................................................... 14  
1.1 WP2 Networks and Communication Services ....................................................... 14  
1.2 Task 2.2 IoT Addressing & Naming for Manufacturing Environments ............ 14  
1.3 Scope of this Deliverable ...................................................................................... 15  
1.4 Structure of the Deliverable .................................................................................. 15  
2 State of the Art Update on IoT addressing and Naming Technologies 17  
2.1 Naming and Addressing Needs of the Internet of Things Network .................... 17  
2.2 Existing Identifiers and Addressing Protocols .................................................... 18  
2.2.1 The IEEE Standards and Media Access Control (MAC) identifiers ............... 19  
2.2.2 Network Layer Internet Protocol (IP) Addresses ............................................. 22  
2.3 Higher Layers Identifiers and Naming Schemes ............................................... 23  
2.3.1 The URN schemes ......................................................................................... 23  
2.3.2 ISO/IEC 9834-1 and Object Identifier (OID) schemes ................................... 24  
2.3.3 EPCglobal ..................................................................................................... 24  
2.3.4 Digital Object Identifier (DOI) names ........................................................... 26  
2.3.5 NetBIOS ....................................................................................................... 28  
2.4 Managing Names and Addresses ....................................................................... 28  
2.4.1 Host Configuration Protocols and Address Discovery .................................... 28  
Dynamic Host Control Protocol (DHCP) and Automatic IP .................................. 29  
Link layer discovery protocol (LLDP) ..................................................................... 29  
Internet Control Message Protocol (ICMP) ........................................................... 30  
2.4.2 Mapping Function Between Link and Network Layers .................................. 31  
Address Resolution Protocol (ARP) ...................................................................... 31  
Neighbour Discovery Protocol (NDP) .................................................................... 31  
6LoWPAN .................................................................................................................. 31  
2.4.3 TCP/IP Network Information Management: SMI, MIB and SNMP ............... 32  
2.4.4 Mapping Network Identifiers to Applications ............................................... 33  
The Real-time Operating System Nucleus Project (TRON): Ubiquitous Code (ucode) and the T-Engine ................................................................. 33  
The Domain Name System and Domain Name Service (DNS) .............................. 33  
Object Name Service (ONS) ................................................................................... 34  
2.5 Section Conclusions ......................................................................................... 34  
3 Application Layer Service Discovery ..................................................................... 35  
3.1 Using Domain Name Service for Service Discovery (DNS - DS) ..................... 35  
3.2 Simple Service Discovery Protocol (SSDP) ....................................................... 35  
3.3 Discovery in P2P networks .................................................................................. 36  
3.3.1 Flooding ........................................................................................................ 36  
3.3.2 The DHT Approach ...................................................................................... 36  
3.3.3 The Napster Approach ................................................................................. 36  
3.3.4 Prospect ...................................................................................................... 37  
3.4 Middleware for Service Discovery ...................................................................... 37  
3.4.1 Zero Configuration Networking (ZeroConf) ............................................... 37
3.4.2 Universal Plug and Play (UPnP) ............................................................... 37

3.5 Section Conclusions .................................................................................. 38

4 Evolution of Directory Services ................................................................ 39

4.1 Definition and Characteristics of Directories ......................................... 39

4.2 Applications for Directories in IoT@Work ............................................. 39

4.3 The X.500 Standard .................................................................................. 40

4.4 LDAP (Lightweight Directory Access Protocol) .................................... 40

4.5 LDAP/X.500 based implementations for directory services ............... 40

4.6 Similar Technologies ............................................................................... 40

4.6.1 Network Information Service (NIS) ...................................................... 40

4.6.2 Universal Description, Discovery and Integration (UDDI) ................. 40

4.6.3 Directory Service Markup Language (DSML) ................................... 40

4.6.4 Meta directories .................................................................................... 41

4.7 Section Conclusions ............................................................................... 41

5 Secure Device IDs ..................................................................................... 42

5.1 Self-Generated Uncertified Secure IDs .................................................. 42

5.2 Self-Generated Certified Secure IDs ....................................................... 43

5.3 Assigned Certified Secure IDs ............................................................... 43

5.4 Assigned Uncertified Secure IDs .......................................................... 44

5.5 Section Conclusions ............................................................................... 44

6 Industrial Configuration Tools .................................................................. 45

6.1 Industrial Engineering Tools and Adopted Protocols ............................ 45

6.1.1 Common Engineering Process of an Industrial Plant ......................... 45

6.1.2 Device Description ............................................................................. 46

6.1.3 Generic Interfaces for Device Management ....................................... 46

6.1.4 Existing Engineering Tools ................................................................. 47

6.1.5 Automation Application Programming Languages (IEC 61131) ....... 47

6.1.6 Startup Phase of an Industrial Controller and Corresponding Communication Protocols ................................................................. 47

6.1.7 Section Conclusions ........................................................................... 49

7 Scenario Driven Requirements on Naming and Addressing ............. 50

7.1 Introduction to the Requirement Methodology ...................................... 50

7.2 Large Scale Manufacturing and Agile Manufacturing Scenarios .......... 50

7.2.1 Cloneable Subsystems ....................................................................... 50

7.2.2 Extensible Subsystems ..................................................................... 50

7.2.3 Company Wide Access .................................................................... 50

7.3 Remote Access Manufacturing ............................................................... 51

7.3.1 Directory entries and data-base management ................................... 51

7.3.2 Dealing with Address Changes ......................................................... 51

7.3.3 Remote Access .................................................................................. 51

7.3.4 Cross-Checking Device Identity ......................................................... 52

8 Future Trends and Needs ........................................................................ 53

8.1 Standardization Activities .................................................................... 53

8.1.1 AutomationML .................................................................................. 53

8.1.2 ISA99 - Industrial Automation and Control System Security ........ 53
8.1.3 ISA 100 standards – Family of wireless standards ........................................ 54
8.1.4 Other related standards ............................................................................. 54

8.2 Automatic configuration and self-organization ............................................. 55
8.2.1 Definition and characteristics of automatic configuration and self-organization 55
8.2.2 Self-organization in automation systems .................................................... 56
8.2.3 Requirements in automation systems ......................................................... 56
8.2.4 Role definition ............................................................................................ 57
8.2.5 Service discovery for self-organization ....................................................... 57
8.2.6 Methods and techniques of self-organization ............................................. 58
IPv6 automatic configuration mechanisms ......................................................... 58
Stateless auto-configuration ............................................................................ 58
Stateful auto-configuration ............................................................................... 58
8.2.7 Design paradigms of self-organization ......................................................... 58
Designing local behavior to achieve global properties ........................................... 58
Tolerating imperfect coordination by implicit coordination ................................. 59
Minimizing long-lived state information ............................................................ 59
8.2.8 Protocols for adaptability to changes .......................................................... 59
Architectural considerations ............................................................................ 59
8.2.9 Limitations of self-organization ................................................................. 59
8.2.10 Future Challenges .................................................................................... 60
Plug & work in automation systems ................................................................. 60
Self-organization based on peer-to-peer overlay networks ................................... 61

9 Conclusions and Next Steps ......................................................................... 62

10 References .................................................................................................... 66
List of Figures

Figure 2.1 Ethernet Address (Figure can be found Wikipedia) ........................................ 22
Figure 2.2 LLDP (Source: Blatherwick, Romascanu, IETF-68, IETF ECRIT / IEEE Joint Meeting) ....................................................................................................................... 30
Figure 3.1: Automatic Configuration and Service Discovery for Networked Smart Devices (see Obiltschnig, 2006 [24]) ............................................................... 38
Figure 5.1: classification of secure IDs ........................................................................ 42
Figure 6.1: General Workflow of Engineering an Industrial Process ......................... 45
Figure 6.2: Assignment of the IP addresses to the IO-Device, normal sequence [36] ................................................................................................................................. 48
Figure 6.3: Profinet AR setup sequence [36] ................................................................. 48

Figure 9.1: IoT@Work Naming and Addressing Function Split Across the Different Layers: (the colour blue indicate the sub-layers where IoT@Work extensions are needed) ......................................................................................... 63

List of Tables

Table 2.1 – ISO reference model for OSI (addressing and address management related protocols) .......................................................................................................................... 19
Table 9.1 – Follow-up Activities Based on This Deliverable .................................... 65
## List of Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
</table>
| ARP     | Address Resolution Protocol  
Is a protocol used in local area networks to translate IP addresses to Ethernet addresses |
| ARQ     | Automatic Repeat reQuest or Automatic Repeat Query  
is an data communication error-control mechanism based on the use of acknowledgements and timeouts to achieve reliable data transmission over an unreliable service |
| BMBF    | Bundesministerium für Bildung und Forschung  
German Ministry for Research and Education |
| BOOTP   | Bootstrap Protocol  
is a network protocol used by clients to obtain an IP address from a configuration server. The DHCP is a more advanced protocol that has superseded the use of BOOTP |
| CCM     | CORBA Component Model  
is part of CORBA and it describes a standard application framework for CORBA components that makes possible to use a network infrastructure to communicate with sensors, controllers, operator terminals and actuators |
| DCOM    | Distributed Component Object Model  
is a Microsoft technology for communication among distributed software components |
| DCS     | Distributed Control System  
Is a process control system using a network infrastructure to communicate with a set of sensors, controllers, operator terminals and actuators |
| DHCP    | Dynamic Host Configuration Protocol  
is a protocol used on IP networks for configuring network devices. There are specific versions of DHCP for IPv4 and IPv6 |
| DPWS    | Devices Profile for Web Services  
defines a minimal set of functionalities to enable secure Web Service messaging, discovery, description, and eventing on resource-constrained devices. It is similar to Universal Plug and Play (UPnP) with the main difference of being aligned with Web Services technology |
| EDDL    | Electronic Device Description Language  
is used by manufacturers to describe the information that is accessible in digital devices |
| FDT/DTM | Field Device Tool/Device Type Manager  
is an open framework for field device tools. FDT defines the data exchange interface between field devices and each of the control systems, engineering tools and asset management system tools. DTM is the software component that works on the framework and facilitates operation through a graphical interface |
| **Foglets** | Foglets or Utility fogs  
is a term, invented by Dr. John Storrs Hall, to describe a collection of cooperating tiny robots to perform a specific function or achieve a specific objective. |
| **HMI** | Human Machine Interface  
The user interface in a manufacturing or process control system. Normally the HMI provides a graphics-based representation of an industrial control and monitoring system. Sometimes it also identified as MMI (man machine interface).  
The HMI typically resides in an Windows based computer that communicates with a specialized computer (e.g. PLC, PAC, DCS) in the plant |
| **IDL** | Interface Description Language or Interface Definition Language  
is a specification language used to describe, in a programming language independent way, a software component's interface.  
IDLs are normally used in remote procedure call software frameworks |
| **IEEE 802** | Institute of Electrical and Electronics Engineers  
The Institute of Electrical and Electronics Engineers is an international non-profit, professional organization for the advancement of technology related to electricity and electronics.  
IEEE 802 refers to a family of IEEE standards dealing with local and metropolitan area networks, e.g. Ethernet (IEEE 802.3) or Wireless LAN (IEEE 802.11) |
| **IETF** | Internet Engineering Task Force  
It is an open standards organization, based on volunteers, that develops and promotes Internet standards, in particular the ones related to the TCP/IP and Internet protocol suite.  
The IETF is formally a part of the Internet Society and is overseen by the Internet Architecture Board (IAB) |
| **IGMP** | Internet Group Management Protocol  
is a communications protocol used on IP networks by hosts and routers to manage multicast group memberships |
| **OLE** | Object Linking and Embedding  
is a Microsoft technology to embed and/or link documents and other objects |
| **OPC** | OLE for Process Control  
It was originally a set of standards specification based on Microsoft OLE structuring the communication of real-time plant data between control devices from different manufacturers.  
Currently the OPC specifications are managed by the OPC Foundation. Since June 2006 OPC is a set of seven standards specifications (and two more in the standardization process). The original standard is currently known as OPC Data Access or OPC Data Access Specification.  
OPC is heavily used not only within the process industries, but also in discrete manufacturing |
| **OPC-A&E** | OPC Alarms & Events  
Provides alarm and event notifications on demand (in contrast to the
continuous data flow of Data Access). These include process alarms, operator actions, informational messages, and tracking/auditing messages

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
</table>
| OPC Batch | OPC Batch  
This spec carries the OPC philosophy to the specialized needs of batch processes. It provides interfaces for the exchange of equipment capabilities (corresponding to the S88.01 Physical Model) and current operating conditions |
| OPC Complex Data | OPC Complex Data  
A companion specification to Data Access and XML-DA that allows servers to expose and describe more complicated data types such as binary structures and XML documents |
| OPC-DA | OPC Data Access  
Used to move real-time data from PLCs, DCSs, and other control devices to HMIs and other display clients. The Data Access 3 specification is now a Release Candidate. It leverages earlier versions while improving the browsing capabilities and incorporating XML-DA Schema |
| OPC-DX | OPC Data eXchange  
This specification takes us from client/server to server-to-server with communication across Ethernet fieldbus networks. This provides multi-vendor interoperability! And, oh by the way, adds remote configuration, diagnostic and monitoring/management services |
| OPC-HDA | OPC Historical Data Access  
Where OPC Data Access provides access to real-time, continually changing data, OPC Historical Data Access provides access to data already stored. From a simple serial data logging system to a complex SCADA system, historical archives can be retrieved in a uniform manner |
| OPC Security | OPC Security  
All the OPC servers provide information that is valuable to the enterprise and if improperly updated, could have significant consequences to plant processes. OPC Security specifies how to control client access to these servers in order to protect this sensitive information and to guard against unauthorized modification of process parameters |
| OPC-UA | OPC-UA  
A new set of specifications that are not based on Microsoft COM that will provide standards based cross-platform capability |
| OPC XML-DA | OPC XML-DA  
Provides flexible, consistent rules and formats for exposing plant floor data using XML, leveraging the work done by Microsoft and others on SOAP and Web Services |
| OpenVPN | OpenVPN  
is a free and open source software application delivered under the GNU General Public License that provides virtual private network (VPN) features for creating secure point-to-point or site-to-site connections. It uses SSL/TLS for encryption and is able to cross network address translators (NATs) and firewalls. OpenVPN allows mutual authentication of peers using pre-shared secret keys, certificates, or username/password |
| ORB | Object Request Broker  
in Common Object Request Broker Architecture (CORBA ORBs manage... |
the transformation (called marshalling/serialization and un-marshalling/de-serialization) of in-process data structures to and from the byte sequence transmitted over the network

<table>
<thead>
<tr>
<th><strong>P2P</strong></th>
<th><strong>Peer to Peer</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>is a networking paradigm usually referring to applications that are programmed in a peer to peer manner rather then based on a client-server approach. The peering entities are equal in the amount of information they manage or the way they act in a large distributed system. P2P protocols are inspired by the above paradigm and can be applied to different layers of the ISO/OSI protocol model.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>PAC</strong></th>
<th><strong>Programmable Automation Controller</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Is a programmable microprocessor-based device used in discrete manufacturing, process control and remote monitoring applications. PACs combine the functions of a PLC with the greater flexibility of a PC, and are able to provide in a single system the functionalities of a DCS and PLC</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>PLC</strong></th>
<th><strong>Programmable Logic Controller</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>A programmable microprocessor-based device used in discrete manufacturing to control assembly lines, machinery or other types of mechanical, electrical and electronic equipment on the shop floor</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>PDA</strong></th>
<th><strong>Personal Digital Assistant</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>A kind of palmtop computer</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>POA</strong></th>
<th><strong>Portable Object Adapter</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>an object adapter is a mechanism to connect a request submitted to an object reference to the code able to service that request. The POA is a particular type of object adapter that is defined in CORBA</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>PROFIBUS</strong></th>
<th><strong>Process Field Bus</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>is a standard for field bus communication in automation technology. There are two variants of PROFIBUS currently in use:</td>
<td></td>
</tr>
<tr>
<td>• PROFIBUS DP (Decentralized Peripherals) is used to manage sensors and actuators via a centralized controller in production (factory) automation applications. This is the most commonly variant;</td>
<td></td>
</tr>
<tr>
<td>• PROFIBUS PA (Process Automation) is used to monitor measuring equipment via a process control system in process automation applications</td>
<td></td>
</tr>
<tr>
<td>The initial version of PROFIBUS, PROFIBUS FMS (Fieldbus Message Specification) was not flexible enough. Even if PROFIBUS FMS is still in use, the majority of users find newer solutions to be useful.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>PROFIBUS GSD</strong></th>
<th><strong>PROFIBUS Generic Station Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Is used to describe device capabilities. These descriptions are normally arranged in files (so called GSD files). With a GSD file, system integrators can determine basic data such as the communications options and the available diagnostics</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>PROFIBUS GSDML</strong></th>
<th><strong>PROFIBUS Generic Station Description XML</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>GSDML is an XML format of GSD</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>PROFINET</strong></th>
<th><strong>PROFINET</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>is the open industrial Ethernet standard for automation. PROFINET uses TCP/IP and IT standards and supports real-time Ethernet communications</td>
<td></td>
</tr>
</tbody>
</table>
| **PROFIsafe** | **PROFIBUS safety or PROFINET safety**
| --- | ---
| PROFIsafe is an open functional safety communication technology for distributed automation systems that integrates safety into the PROFIBUS/PROFINET fieldbus technologies as a separate layer on top of the fieldbus application layer and reduces the error probability of the data transmission to the level required by the relevant standards. PROFIsafe can be used in safety applications up to: Safety Integrity Level 3 according to IEC 61508, Performance Level "e" according to ISO 13849, or Category 4 according to EN 954-1. 
PROFIsafe is standardized in IEC 61784-3-3 |
| **REST** | **Representational State Transfer**
| --- | ---
| REST is a software architecture for distributed systems in the context of HTTP (even if it can be based on other applications protocols that provide support for meaningful resources representational states) centered around the transfer of representations of resources, where a resource is potentially any coherent concept that can be addressed and a representation is normally a set of information that capture the state of the corresponding resource.
A system or service REST complaint is referred to as a RESTful system/service. |
| **SCADA** | **Supervisory Control And Data Acquisition**
| --- | ---
| SCADA refers to industrial control systems: computer systems that monitor and control industrial processes or infrastructures |
| **SOA** | **Service Oriented Architecture**
| --- | ---
| SOA Provides methods for systems development and integration where systems group functionality around business processes and package these as interoperable services. An SOA infrastructure allows different applications to exchange data with one another as they participate in business processes. Service-orientation aims at a loose coupling of services with operating systems, programming languages and other technologies which underlie applications |
| **UPnP** | **Universal Plug and Play**
| --- | ---
| UPnP is a set of networking protocols finalised to enable networked devices to discover other network devices and the services these provides. The UPnP technology is promoted by the UPnP Forum. |
| **UID** | **Unique Identifier**
| --- | ---
| **W3C** | **World Wide Web Consortium**
| --- | ---
| W3C is the main international standards organization for the World Wide Web. Amongst others, it is responsible for the XML standardization |
| **WPAN** | **Wireless Personal Area Network**
| --- | ---
| **WS&AN** | **Wireless Sensor and Actuator Network**
| --- | ---
| WS&AN refers to a group of sensors and actuators linked by wireless communication facilities to perform distributed sensing and actuation tasks |
1 Introduction

1.1 WP2 Networks and Communication Services
Plug&Work will simplify integrating devices in the network, relying on self-organizing techniques and protocols. The orchestration between available heterogeneous networking resources and the application demands is the focus of this WP. Both in a setup phase and during operation, allocation of network resources connecting and supporting IoT applications will have to occur in transparent manner to the application supporting the idea of invisible networks. Whereas the focus of this WP is on realizing “advanced communication services” that are offered to automation and factory-based applications, through a well defined interface. The interface definition should be both applicable to IoT application interactions and data-driven mashups in a later stage, since most of the automation scenarios are rather static. Adapting the advanced communication services and the system-wide resource orchestration to change is driven by the needs of Plug&Work IoT. The development of this orchestration system as well as the interface extensions to support full Plug&Work will be carried out in an iterative and adaptive manner (agile development), and in close synchrony with the updates and finding of the WP1 (architecture and system requirements), and the evaluation of the security aspect of the system, carried out in WP3. The result of the WP will be a validated and integrated framework for Plug&Work and secure networks for IoT-based automation systems in two parallel tasks (Task 2.5 and Task 3.6). Objectives of addressed by the WP include:

- Supporting Plug&Work at all levels of automation systems, from the application level down to the plug and play network
- Providing communication services for industrial environments on-demand and in an adaptive manner, over a shared heterogeneous communication infrastructure, and catering for very different application needs
- Developing self-organization / self-configuration concepts and protocols to provide the required robustness, reliability and real-time support for industrial environments; only minimal network configuration effort in the communication network nodes should be required for setting up a fully functional industrial communication network
- Enhance flexibility for the advanced communication infrastructure supporting different industrial services and applications with different requirements
- Enable automated network management with as few manual configuration as possible
- Comparison of centralised and decentralised network management approaches
- Take care of mobility requirements in industrial environments by introducing reliable, robust and real-time wireless links using the newest wireless network technologies.

1.2 Task 2.2 IoT Addressing & Naming for Manufacturing Environments
This task will integrate mechanisms for naming and addressing based on IoT existing developments. The generated name and address space will target traditional problems of separating locator and identifiers from the name or address scheme, where approaching both problems according to the needs and requirements found in the factory field.
The addressing problem requires, in state of the art factory planning tools, a complex and costly manual management effort of matching MAC addresses to IP addresses, and to device names. The goal of this task is to investigate existing self-configuration protocols and analyzing needs for additional extensions such as mapping address spaces based on multi-dimensional transformations, where each dimension such as, device unique ID, security credentials, network neighborhood, and semantic role of device in the application scope (e.g. an automation unit or cell can be defined as a boundary to define the scope), all can make part of the addressing scheme.

The use of decentralised mechanisms and protocols, such as distributed hash tables and other P2P concepts, will also be investigated. The integration of semantic information related matching the physical world with the virtual world is one contribution of this work.

The self-organizing system should also support discovery of devices and capabilities and include this in the naming and localisation mechanism. Both a global Internet support for device and service localisation, as well as intranet use cases should be supported by the proposed framework.

The naming and addressing framework will take Plug&Work requirements into account, especially dealing with device exchange, failure, location change (due to mobility), or service migration (shifted virtual process). The results of this task will be provided in deliverable D2.1.

### 1.3 Scope of this Deliverable

The deliverable D2.1 is the first step towards the definition of a bootstrapping mechanism that implements the IoT@Work proposed Plug and Work approach. For this purpose the deliverable looks first at the enabling technologies found in the Internet and more specifically the latest developments towards the Internet of Things (IoT). The deliverable will enlist the defined IoT addressing and naming schemes that could be integrated in IoT@Work. Ideas related to service and application semantics are defined. The requirements of Plug&Work are investigated with the help of the scenarios provided in Deliverable D1.1 (State of the Art and Initial Requirements). The naming and addressing requirements of the three scenarios are detailed in this document as well.

### 1.4 Structure of the Deliverable

The deliverable is structured in the following way. Section 2 starts by making an analysis of the naming and addressing problematic in the Internet. A focus is made on developments made to enable the Internet of Things. Industrial and factory automation adopted protocols for naming and addressing are explained in more details. Section 3 presents service discovery mechanisms existing in higher layers and that are often integrated in middleware approaches. The goal of the section is to analyze the role of middleware for embedded and distributed systems in the setup of self-organizing system. The use of directories for both names, service instances, and resources are also an important element of architecting the IoT, therefore Section 4 looks into the evolution of directory systems. The need for a secure bootstrap and the existing security mechanisms are detailed in Section 5. In Section 6, configuration tools are introduced to explain the current practice in configuring names and addresses in factory automation systems. Section 7 analyses the needs of auto-configuration, locator and identifier separation problems and indirection needs identified in the IoT@Work scenarios.
Section 8 provides a list of trend analysis and current developments of state of the art that could affect both the bootstrapping architecture and tooling and or implementation details of proposed functionalities. The conclusion of the deliverable together with the next steps and that will use this deliverable are summarized in Section 9.
2 State of the Art Update on IoT addressing and Naming Technologies

The below sections are related to the state of the art collections as found in the literature and previous projects run on the enablement of the Internet of things. The goal of this review is to illustrate the different solutions existing out there. IoT@Work’s focus on ease of configuration and use of names and addresses will be based on a deep understanding of state of the art and the existing solutions. The resulting bootstrapping architecture will use as many of the existing solutions as possible rather trying to invent new protocols.

2.1 Naming and Addressing Needs of the Internet of Things Network

The ability to uniquely identify items, devices, locations and services is essential for the Internet of Things. The purpose of identification (naming) is to map a unique identifier (name) or UID to an entity so as to make it without ambiguity identifiable and retrievable. UIDs may be built as a single quantity or out of a collection of attributes such that the combination of their values is unique. In the vision of the Internet of Things, things have a digital identity (described by unique identifiers) and the relationships among things can be specified in the digital domain.

Identification schemes and technologies must be able to handle the following issues:

- **Uniqueness**: Identifiers must be unique. A unique identifier is an identifier assigned to at most one object in the universe (for globally unique identifier or GUID) or within a particular scope.

- **Consistency**: An identifier must have the same meaning everywhere. Context should not influence the meaning or interpretation of an identifier.

- **Persistence/Longevity**: The lifetime of an identifier should ideally be the same as the lifetime of the object it identifies. The lifetime of GUID is expected to have no limits, that is, the identifier may be used to identify an object well beyond the lifetime of the object it identifies.

- **Scalability and Extensibility**, where the system boundary is no longer limited to the boundary of a machine or a device, but can be extended by adding a single or multitude of devices and services that are easily integrated at physical level and applications levels. Such a target makes identifying each component in the system an important requirement. However, it also creates scalability and complexity problems that have to be taken care of.

- **Identity Management**:
  - Multiple identifiers per object and possible cross-referencing among identifiers for the same entity.
  - Allow changes of identifier, changes of configuration and association between identifiers to be recorded and queried e.g. parent-child (composite objects) and old-new (part replacement) relationships.

- The encoding of an identifier should be capable of being communicated easily over any communication interface irrespective of the data format or the access technology used.

- In some cases it is useful to be able to transparently modify ("translate") identifiers to create structured name spaces or to create routing/security domains.
• Support various existing and future identifier schemes. Interoperability is required between applications using different schemes.

• When identifiers from multiple identification schemes are to be used on different objects within the same application or set of applications, the identification schemes themselves must be uniquely identifiable to enable the correct interpretations of the objects.

• Secure IDs: Having an inherent association between identifiers and security credentials (or the identifier itself being a security credential) eases credential management (e.g. during bootstrapping of security) and supports security services of all kind.

The ability to uniquely identify objects is useful at different value chains:

• Supply-Chain Management: RFIDs for product memory, product life cycle management (where is the product at each time, with which status).

• Internet of Things for machine2machine communication, how to form a device symbolic name that maps its semantic description

• Mapping Communication: Mapping from a symbolic device name to its semantic description (including address) to enable Internet of machine-to-machine in a Internet Of Things network.

• Resource discovery. Identifiers play a critical role in the ability of a system to discover distributed resources such as web services.

In the context of the Internet Of Things in manufacturing environments, naming and addressing technologies have to take into consideration the presence of advanced communication networks that host state of the art planning and automation tools which require real-time response. Furthermore, they should also facilitate Plug&Work, that is, the ability of devices and network components to auto-configure themselves according to the needs of the automation applications. Currently, a complex and manual management effort is required in matching MAC addresses to IP addresses and device names. A third important requirement from IoT is a name space scaling to a very high number of identifiable objects.

Thus, the naming and addressing framework must:

• Take Plug&Work requirements into account, especially dealing with device exchange, failure, location change, or service migration.

• Integrate the semantic information related to the matching of the physical world with the virtual world, where application names and semantic roles of the devices influence the mapping between application level functionality, services and MAC and IP addresses.

### 2.2 Existing Identifiers and Addressing Protocols

This section will review the established protocols dealing with locators and identifiers at the different layers of the ISO/OSI layer stack. The layers taken into account are Physical and Link layer, then network layer, and finally the application or service layers.

Out of the physical and link layer protocols that have brought Internet technologies into factory floor networks are Ethernet protocols. This meant that similar to local area networks (LANs) and wide area networks (WANs) similar address conventions exist at the factory field. In this section, the address schemes adopted in those layers are explained.
<table>
<thead>
<tr>
<th>OSI layer</th>
<th>Protocols, IPv4</th>
<th>Protocols, IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>DNS (Domain Name System), SNMP (Simple Network Management Protocol), DHCP (Dynamic Host Configuration Protocol), SSDP (Simple Service Discovery Protocol)</td>
<td>DHCPv6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SSDPv6</td>
</tr>
<tr>
<td>Presentation</td>
<td>i.e. SIP (Session Initiation Protocol)</td>
<td></td>
</tr>
<tr>
<td>Session</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transport</td>
<td>TCP/UDP</td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td>HIP (Host Identity Protocol), Mobile IP ICMP (Internet Control Message Protocol) NAT (Network Address Translation) various tunneling methods (i.e. IP-in-IP)</td>
<td>ICMPv6</td>
</tr>
<tr>
<td>Data link</td>
<td>LLDP (Link Layer Discovery Protocol) ARP (Address Resolution Protocol)</td>
<td>NDP (Network Discovery Protocol)</td>
</tr>
<tr>
<td>Physical</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1 – ISO reference model for OSI (addressing and address management related protocols)

2.2.1 The IEEE Standards and Media Access Control (MAC) identifiers

**IEEE 802.** The IEEE 802 refers to the family of IEEE standards restricted to variable-sized packet networks. They deal with Local and Metropolitan Area Networks (LAN and MAN, respectively). The most widely used standards are for the Ethernet family, Token Ring, Wireless LAN, Wireless PAN, Wireless MAN, Bridging and Virtual Bridged LANs. IEEE 802 does not deal with cell relay or isochronous networks.

The set of services and protocols considered by IEEE belong to the Data Link and Physical layers of the OSI reference model. Below, we include the standards that will most likely be associated with an industrial environment.

- **IEEE 802.1 High Level Interface (Internetworking):** describes procedures for packet forwarding and forming of domains. Important protocols include:
  - 802.1D – Spanning Tree Protocol
- 802.1p – Traffic Class Expediting and Dynamic Multicast Filtering (published in 802.1D-1998)
- 802.1Q – Virtual Bridged LANs
- 802.1s – Multiple Spanning Tree Protocol
- 802.1w – Rapid Spanning Tree Protocol
- 802.1X – Port Based Network Access Control
- 802.1AB – Link Layer Discovery Protocol

- IEEE 802.2 (Logical Link Control) - amongst others, defines Quality of Service in IEEE 802 networks
- IEEE 802.3 (Ethernet). Standards that define the physical layer and data link's layer MAC of wired Ethernet
- IEEE 802.11 (WLAN). Standards that define a wireless physical layer and the medium access control (data link layer) for wireless local area networks
- IEEE 802.15 (WPAN). Wireless PAN (Personal Area Networks)
  - IEEE 802.15.1 - WPAN based on Bluetooth.
  - IEEE 802.15.4 - Low Rate WPAN. Wireless PANs based on low-cost and low speed ubiquitous communication between devices. It forms the basis for several proprietary network (or mesh) protocols such as ZigBee, 6LoPAN, WirelessHART and MiWi. TinyOS, Unison RTOS and DSPnano RTOS stacks also use a few items of IEEE 802.15.4 hardware.
    Furthermore, IEEE 802.15.4e expands/amends 802.15.4 MAC to better support industrial markets e.g. Factory Automation, Process Automation.
  - Industrial Ethernet Protocol and Technologies: ModBus/TCP, EtherCAT (Ethernet for Control Automation Technology), PROFINET RT and IRT, Ethernet Powerlink, EtherNet/IP.

A Media Access Control identifier, also referred as MAC address, is a unique identifier assigned to network interfaces for communications, such as a network card, on the physical network segment. Logically, MAC addresses are used in the Media Access Control protocol sub-layer of the OSI reference model.

A MAC address may also be known as an Ethernet hardware address (EHA), hardware address, adapter address, or physical address. MAC addresses are most often assigned by the manufacturer of device e.g. a network interface card (NIC), and are stored in its hardware, the device's read-only memory, or some other firmware mechanism. If assigned by the manufacturer, a MAC address usually encodes the manufacturer's registered identification number or Organisation Unique Identifier (OUI), which is a 24-bit company-id value administered and assigned by the IEEE Registration Authority. Currently there exist three address spaces for MAC addresses, namely, MAC-48, EUI-48 and EUI-64.

A MAC-48 identifier consists of two partitions, the 24-bit OUI and a 24-bit number assigned by the owner of the OUI according to the ANSI/IEEE 802 standard. MAC-48 addresses are used by Ethernet, 802.11 wireless networks, Bluetooth, IEEE 802.5 token ring (and most other 802 networks), FDDI and ATM.

Extended Unique Identifiers (EUI-48 and EUI-64) are used to identify software and hardware devices. EUI-48 and MAC-48 addresses are syntactically indistinguishable. EUI-48 is intended to replace MAC-48. In fact, the IEEE now considers the label
MAC-48 to be an obsolete term which was previously used to refer to a specific type of EUI-48 identifier used to address hardware interfaces within existing 802-based networking application. EUI-64 addresses were introduced in order to solve the MAC address exhaustion problem. A EUI-64 address is made of two partitions, the 24-bit OUI and a 40-bit organisation-specific identifier. EUI-64 identifiers are supported in Firewire, IPv6, ZigBee/802.15.4/6LoWPAN [72] and [77].

**Individual Address Blocks (IABs)** are EUI-48 addresses that were created to allow organizations to purchase smaller blocks of identifiers at a reduced price. An IAB is made of the 24-bit OUI prefix followed by an additional 12-bit extension identifier (also assigned by the IEEE Registration Authority), thus leaving additional 12 bits for use by the assignee. The resulting 48-bit identifier uniquely identifies the assignee of the IAB and provides 4096 (2^12) unique EUI-48 numbers for use by the organization that purchased the IAB.

**Virtual and private MAC addresses** can be used for assigning private MAC addresses to virtual systems (e.g. a virtual machine in the form of VMware emulates a whole computer system with its own virtual network interface card). VMware assigns a virtual MAC address to the different virtual NICs sharing the same physical NIC.

The MAC address is used to classify packets within a context. If you share an interface, but do not have unique MAC addresses for the interface in each context, then the destination IP address is used to classify packets. The destination address is matched with the context NAT configuration, and this method has some limitations compared to the MAC address method. See the *Cisco ASA 5500 Series Configuration Guide using the CLI* for information about classifying packets.

In the rare circumstance that the generated MAC address conflicts with another private MAC address in the network, the MAC address can be setup manually for the interface within the context.

Another use of virtual MAC addresses is for setting up virtual private LAN service (VPLS). The latter is a way to provide Ethernet based multipoint to multipoint communication over IP/MPLS networks. It allows geographically dispersed sites to share an Ethernet broadcast domain by connecting sites through pseudo-wires.

---

1 See VMWare recommendations for using virtual MAC addresses: http://www.vmware.com/support/ws55/doc/ws_net_advanced_mac_address.html

Since VPLS links multiple Ethernet broadcast domains together, it effectively creates a much larger broadcast domain. Since every provider edge (PE) router must keep track of all MAC addresses and associated LSP routing information, this can potentially result in a large amount of memory being needed in every PE in the mesh. To counter this problem, sites may use a router as the customer edge (CE) router. This hides all MAC addresses on that site behind the CE's MAC address. PE devices may also be equipped with content-addressable memory (CAM), similar to high-end Ethernet switches. An alternative mechanism is using MAT (MAC Address Translation). However, at the time of writing this, there aren't vendors providing MAT functionality.

The Broadcast address is a special address ("FF:FF:FF:FF:FF:FF" in MAC-48) which is used to send a packet to all reachable targets within one Ethernet domain.

A Multicast address is received by all stations on a LAN that have been configured to receive packets sent to that address. It has a value of 1 in the least-significant bit of the first octet. Ethernet multicast addresses are typically used as an efficient mapping for IP-layer multicasts (or anycasts in IPv6).

### 2.2.2 Network Layer Internet Protocol (IP) Addresses

IP addresses were designed to be used with the IP packet routing service. A higher-level communication protocol designed to enable systems on different connected networks to communicate. One of the primary application specific design requirements for IP addresses was to enable routers to quickly distinguish between different logical networks. This requirement led IP designers to structure the IP address in a hierarchically encapsulated manner that enables the router to quickly identify network addresses.

The Internet Protocol version 4 (IPv4) is the fourth revision in the development of the Internet Protocol (IP) and it is the first version of the protocol to be widely deployed (RFC 791). Together with IPv6 [76], it is at the core of standards-based internetworking methods of the Internet. IPv4 is still by far the most widely deployed Internet Layer protocol. As of 2010, IPv6 deployment is still in its infancy.
An IPv4 address is a 32-bit identifier. The initial design of IPv4 addresses uses the first n bits of an IP address to identify the class of the IP address. The value of n ranges from 1 to 4, depending on the class of the address – 0 (class A), 10 (class B), 110 (class C), 1110 (class D) and 1111 (class E). The next m bits are used to identify the individual network and depend upon the class of address. The remaining 32-n-m bits are structured in a network and host part depending on the network mask.

The notion of address classes has become less important with the development of classless addressing. Classless Inter-Domain Routing (CIDR) was designed to permit repartitioning of any address space so that smaller or larger blocks of addresses could be allocated to users.

The IPv4 address scheme does not provide for extensibility (i.e. improved support for options/extensions). This deficiency is addressed in IPv6.

An IPv6 address [76] is a 128-bit identifier divided as follows: 3 bits for Format Prefix (001, FP), 13 bits for Top-Level Aggregation Identifier (TLA ID), 8 bits reserved for Future use (RES), 24 bits for Next-Level Aggregation Identifier (NLA ID), 16 bits for Site-Level Aggregation Identifier (SLA ID), and 64 bits for the Interface Identifier.

In the context of TCP/IP, the combination of the IP address and ports are used to differentiate applications at a given host. The pair IP/Port is also used for address translation (NAT/PAT).

6lowPAN (RFC 4944), which stands for IPv6 over Low Power wireless Personal Area Networks, provides a series of protocols that interface between IPv6 addresses and IEEE 802.15.4 networks.

### 2.3 Higher Layers Identifiers and Naming Schemes

#### 2.3.1 The URN schemes

The Uniform Resource Name [89] system is considered fundamental for the development of the Internet of Things. The purpose of a URN is to provide a globally unique, persistent identifier used for recognition, for access to characteristics of the resource or for access to the resource itself. URNs should satisfy a series of requirements: global scope, global uniqueness, persistence, scalability and legacy supporting (as long as they satisfy the requirements above), extensibility, and independence (name issuing) and resolution i.e. there must be a feasible mechanism to translate a URN to a URL (if the latter exists). URNs are, thus, globally unique and persistent. However, the resources to which they are assigned may not be unique or of permanent existence.

URNs form part of the envisaged Internet Information Infrastructure Architecture (IIIA). The IIIA architecture is composed by:

- **URN**: Identifies a resource or unit of information,
- **URNs**, used for identification,
- **Uniform Resource Locators (URL)**: identifies the location or container for an instance of a resource identified by a URN. The resource identified by a URN may reside in one or more locations at any given time, may move or may not be available at all.
- **Uniform Resource Characteristics (URC)**: Set of meta-level information about a resource e.g. owner, encoding, access restrictions, cost, etc.
The requirements for URNs fit with the overall architecture of Uniform Resource Identification (URI) [83]. The URN has a number of sub-schemes registered under a URI namespace, using a namespace ID (NID). This list includes:

- ISSN: International Serial Standard Number (NID 3, RFC3044)
- OID: Object Identifiers (NID 4, RFC3061)
- NBN: National Bibliography Numbers (NID 10, RFC3188)
- UUID: Universally Unique Identifiers (NID 18, RFC4122)
- NFC: Near Field Communication Forum (NID 26, RFC4122)
- EPC: Electronic Product Code (NID 34, RFC5134)
- Epcglobal: EPCglobal (XML schema and namespace (NID 35, RFC5134)

The URN syntax, specified in RFC 2141, determines a hierarchical structure which needs to be converted into a syntactical format for processing through a set of services residing on the internet implementing name resolution and discovery. Some of the URN schemes listed above do not have a specified resolver, and many do not have an established discovery service.

All URNs must have the following syntax:

\[
\text{<URN> ::= \text{"urn:" <NID> \":\" <NSS>}}
\]

where <NID> is the Namespace Identifier, and <NSS> is the Namespace specific string. For example:

\[
\text{urn:epc:id:sgtin:0614141.012345.62852}
\]

is a URN that belongs EPC URN namespace. The general rules for the syntax of namespace specific codes are also specified in RFC 2141.

### 2.3.2 ISO/IEC 9834-1 and Object Identifier (OID) schemes.

OID is a URN scheme used by a wide variety of computer applications and systems, including ISO applications, X.500 based directories (such as LDAP), SNMP, and communication systems such as H.323. OID schemes have three roots: ITU-T, ISO (including ISO/IEC), and joint ITU-T ISO structures.

ISO/IEC 9834-1 is a RFID standard developed by ISO that takes into account the legacy bar coding systems and the associated identification codes.

### 2.3.3 EPCglobal

EPCglobal is leading the development of industry-driven standards for the Electronic Product Code™ (EPC) and is a subsidiary of the non-profit standards organisation GS1 supporting the use of Radio Frequency Identification (RFID) and related industry-driven standards that should enable accurate, immediate and cost-effective visibility of information throughout the supply chain.

The Electronic Product Code (EPC) is designed as a universal identifier that provides a unique identity for every physical object anywhere in the world, for all time. EPCglobal’s main vision is the universal, unique identification of individual items using EPCs encoded in inexpensive RFID tags.

The EPCglobal Framework Architecture [86] is a collection of interrelated hardware, software, and data standards (EPCglobal standards), together with shared network
services that are operated by EPCglobal, its delegates, and third party providers, all in service of a common goal of enhancing business flows and computer applications through the use of EPCs.

The EPCglobal Network is a community of trading partners engaged in the capture, sharing, and discovery of EPC related data that use the EPCglobal Architecture Framework. The EPCglobal Network is made of the following components:

- **Electronic Product Code (EPC)**
- **ID System**, which consists of EPC tags and EPC readers.
- **EPC Middleware**, which manages real-time read events and information, provides alerts, and manages information for communication to EPC Information Services (EPC IS) and a company’s other existing information services.

Discovery Services e.g. Todo: EPC global XML schema and name space. ONS.

- **EPC Information Services (EPC IS)**: Enables users to exchange EPC-related data with trading partners through the EPCglobal Network.

The EPC structure is defined in the EPCglobal Tag Data Standard [87]. The primary representation of an called the Pure Identity EPC URI, a URI intended for use when referring to a specific physical object in communications within business applications. The EPC URI may also be used at the data capture level when the EPC is to be read from an RFID tag or other data carrier, in a situation when the additional ‘control’ information present on a RFID tag is not needed. The EPCglobal Tag Data Standard also defines two additional representations, the EPC Tag URI and Binary Encoding. The EPC Tag URI denotes a specific URI together with additional control information that is used to guide the process of data capture from RFID tags. The EPC’s Binary Encoding a compact binary format suitable for encoding the same information stored in EPC Tag URIs. The EPCglobal Tag Data Standard defines conversion rules between all representations while the Tag Data Translation Standard provides a machine-readable form of these rules.

EPC is designed as a flexible framework. This allows existing naming structures to be incorporated into the EPC system that can support many existing coding schemes, including many coding schemes currently in use with barcode technology. EPC identifiers currently support 7 identification keys from the GS1 system (SGTIN, SSCC, GIAI, etc) of identifiers, as well as a General Identifier and EPC identifiers that can be used for encoding supplies to the US Department of Defence.

The various concrete representations of the EPC use a system of headers (textual or binary according to the representation) to distinguish one identity scheme from another. The heard is always considered to be part of the ECP, not something apart. For example, the Pure Identity EPC URI

```
urn:epc:id:sgtin:0614141.012345.62852
```

denotes a GS1 SGTIN code.

EPCs are not designed exclusively for use with RFID data carriers. They can indeed be constructed based on reading of optical data carriers, such as linear bar codes and two-dimensional bar codes, such as Data Matrix symbols. The ‘pure identity URI’ canonical representation of an EPC is agnostic to the data carrier technology that was used to attach the unique identifier to the individual physical object.

The **Object Name Service (ONS)** [88] is a service that, given an EPC, can return a list of network accessible service endpoints that pertain to the EPC in question. ONS
does not contain actual data about the EPC. It only contains the network address of services that contain the actual data. ONS is also authoritative in that the entity that has change control over the information about the EPC is the same entity that assigned the EPC to the item to begin with. For example, in the case of an SGTIN EPC, the entity having control over the ONS record is the owner of the SGTIN manager number.

The EPCglobal Network leverages (or intends to) existing Internet technology and infrastructure as much as possible. As such, it adheres to the “hour glass model” of the Internet by standardising on one identifier scheme - namely, EPC - and ensures that it is compatible with the Internet going forward by encoding EPCs as URIs. Furthermore, ONS uses the Internet’s existing Domain Name System (DNS) for looking up (resolving) information about an EPC. This means that the query and response formats adhere to the DNS standards, meaning that the EPC will be converted to a domain-name and the results must be a valid DNS Resource Record.

ONS is implemented using DNS with Naming Authority Pointer (NAPTR). A major disadvantage is the missing authorization and authentication connected to ONS queries. Moreover there information that is stored is mainly on class level like the product type. Finally, ONS can be used to associate EPCs with web services and provides static and dynamic services that can be linked to the EPC.

The EPC Network may contain one or more EPC Discover Services; search engines for EPC related data. A Discovery Service returns locations that have some data related to an EPC. Unlike ONS, in general a Discovery Service may contain pointers to entities other than the entity that originally assigned the EPC code. Hence, Discovery Services are not universally authoritative for any data they may have about an EPC. It is expected that there will be multiple competitively run Discovery Services and that some of them will have limited scope (regional, facility wide, etc).

### 2.3.4 Digital Object Identifier (DOI) names

The Digital Object Identifier (DOI) System [84] is a managed system for persistent identification of content on digital networks. A DOI name is the unique identifier used to identify the physical, digital or abstract entities managed by the DOI System. A DOI name is not intended as a replacement for other identifier schemes or other commonly recognised identifiers: if an object is already identified with another string, the character string of the other identifier may be integrated into the DOI name syntax, and/or carried in DOI metadata for use in DOI applications.

A DOI name is the string that specifies a unique object within the DOI system. The DOI syntax (standardised as ANSI/NISO Z39.84-2005) is made up of a “prefix” element and a “suffix” element separated by a forward slash. There is no defined limit on the length of the DOI name, of its prefix or its suffix elements. The DOI name is case-insensitive and may incorporate any printable characters from the Unicode Standard. For example, the DOI name “10.1000/123456” is made of a prefix element “10.1000” and a suffix element “123456”.

The DOI prefix has two components: A Directory indicator followed by a Registrant code, separated by a full stop (period) e.g. “10.1000”. The Directory indicator is always “10” and distinguishes the entire set of character strings (prefix and suffix) as DOIs within a wider Handle System for resolution. The registrant code is a unique alphanumeric string assigned to an organization that wishes to register DOI names through a DOI registration agency (a registrant may have multiple-registrant codes).

The DOI suffix may be a sequential number, or it may incorporate an identifier generated for or based on another system used by the registrant e.g. ISBN, ISSN,
ISTC. In such cases, the existing system may specify its own preferred construction for such a suffix. For example, the DOI name “10.1038/issn.0028-0836” uses an ISSN suffix.

When displayed on screen or in print, a DOI name is normally preceded by a lowercase “doi:” as shown in “doi:10.1006/jmbi.1998.2354”. The use of the “doi:” prefix follows URI standards as, for example, “ftp:” and “http:”

DOI names can be represented in other forms and schemes. For example, they can be represented in a URL and transported by the HTTP protocol - provided that the DOI name follows the URI syntax (which is more restrictive) – or represented using the INFO URI scheme.

The DOI System provides for unique identification, persistence, resolution, metadata, and semantic interoperability of content entities - it can be used to identify physical, digital, or abstract “objects”. It has grown to be the predominant method for identifying digital media, particularly electronically available reports and papers. The DOI System provides a ready-to-use system of the following several components: a specified numbering syntax, a resolution service (based on the Handle System), a data model system (including the index Data Dictionary), and policies and procedures for the implementation of DOI names through a federation of Registration Agencies.

- A syntax specification, defining the construction of DOI names.
- A resolution component (based on the Handle System), providing a mechanism to resolve the DOI name to data specified by the registrant.
- A metadata component, defining an extensible data model for associating descriptive and other elements of data with the DOI name. An object associated with a DOI name is described unambiguously by DOI metadata.
- A social infrastructure, a set of policies and procedures for the implementation of DOI names through a federation of registration agencies.

A DOI name can, within the DOI system, be resolved to values of one or more types relating to the object identified by that DOI name (multiple resolution) e.g. URL, e-mail, and descriptive metadata. These values may or may not be directly accessible in the form of a digital file or other manifestation; hence the resolution may or may not return an instance of an object. These values can form the basis for other services e.g. a specific local copy of an article can be determined by combining the resolution result (several URLs) and local information about the user’s location.

The Handle System, the resolution component used in the DOI system, is a general-purpose distributed information system [85] designed to provide an efficient, extensible, and secure global name service for use on networks such as the Internet. The Handle System is part of the wider Digital Object Architecture [85] ; an architecture that deals only with digital objects with identifiers (Handles). The Handle System is used in a variety of applications such as the Content Object Repository Discovery and Resolution Architecture (CORDRA) of the US Department of Defence; The Library of Congress National Digital Library Program; and applications in grid computing and advanced future

The DOI system is one implementation of the Handle System: a DOI name is a Handle. The Handle System provides a general-purpose global name service enabling secure name resolution over the internet, designed to enable a broad set of communities to use the technology to identify digital content independent of location. The DOI System utilises the Handle System as one component in building an added
value application, for the persistent, semantically interoperable, identification of intellectual property entities.

The DOI System has grown to be the predominant method for identifying digital media, particularly electronically available reports and papers. It has been developed and implemented in a range of publishing applications since 2000; by early 2009 over 40 million DOIs had been assigned. In particular, some DOI applications have become useful as a de facto registry of a particular content type. Examples are CrossRef (scholarly articles) and DataCite (scientific data sets).

2.3.5 NetBIOS

The Network Basic Input/Output System (NetBIOS) provides services related to the Session layer of the OSI model allowing applications on separate computers to communicate over a local area network. NetBIOS defines a software interface, not a protocol. NetBIOS over TCP/IP (NBT or NetBT) is the networking protocol that implements the NetBIOS API for TCP/IP networks (RFC 1002).

NetBIOS allows browsing of network resources and handles basic networking functions by using a two-way acknowledged data transfer. NetBIOS applications employ NetBIOS mechanisms to locate resources, establish connections, send and receive data with an application peer, and terminate connections.

NetBIOS resources are referenced by name. The name space is flat and uses 16 alphanumeric characters or octets. An application, representing a resource, registers one or more names that it wishes to use. Registration is a bid for use (exclusive or shared) of a name, where each application contends with the other applications in real time. Assignment of names is distributed and highly dynamic.

Under NetBIOS, each computer in the network has both an IP address and a NetBIOS name corresponding to a (possibly different) host name. A computer’s NetBIOS name is often the same as that computer’s host name, but it may also be completely different. In order to connect to a computer running TCP/IP via its NetBIOS name, the name must be resolved to a network address (usually an IP address). This is done by either broadcast or a NetBIOS Name Server e.g. WINS.

2.4 Managing Names and Addresses

Managing names and addresses can be defined from the functional point of view as those functions needed to assign, store, lookup or query the already assigned names and addresses. The above-mentioned functions are implemented totally differently at protocol layer of the ISO/OSI model, because each layer has a totally different scope. For resolving/looking up application instances of the World Wide Web names (web addresses) a totally different mechanism is needed than that of resolving MAC addresses in a local area network. The first example has a global scope, whereas the second a very local one. The solutions that can store, support querying and lookup of the address or name are implemented differently. These protocols are explained next.

2.4.1 Host Configuration Protocols and Address Discovery

The first type of functionality dealt with is the host or node discovery and that takes place when a node joins the network. We apply the convention of computer networks that are called hosts of different services and applications that communicate with each other. The bootstrapping process (i.e. from the moment of adding a physical node to the system until the application is ready to run) starts at the lowest layers, for each protocol entity to be instantiated in its own scope.
The node or host discovery process relies on a series of protocol interactions that complement each other. In this subsection we concentrate on the lower layers (link/network layers).

**Dynamic Host Control Protocol (DHCP) and Automatic IP**

In most LAN networks, hosts receive an IP address once they are connected to the network and are usually automatically assigned an IP address. DHCP and auto-IP configuration are the way to do this. The two protocols are network-layer protocols (at layer 3). Auto-IP is not as spread as DHCP. The latter offers a multitude of options and possibilities to assign IP addresses to hosts. There are also a multitude of possibilities to filter the operation, such as using some pre-configured host identifier and map it to an IP address (e.g. creating a table of (IP,MAC) address tuples).

DHCP uses a Server-Client communication model, where any pre-configuration filters and rules can be specified at the server. This differs to the IPv6 development towards a peer-to-peer model. The automatic IP address assignment, duplicate address detection, and neighbor discovery protocols in IPv6 represent the full extent of the auto-configuration protocols in IPv6. A DHCPv6 protocol development still however exists.

**Link layer discovery protocol (LLDP)**

Connectivity in Ethernet-based LANs is discovered using the LLDP protocol. The latter is the IEEE standard 802.1AB-2005 from May 2005, which allows a network device (IEEE 802 device) to advertise its identity and capabilities on the local network using layer 2 mechanisms. It enables a LLDP agent to learn higher layer management information such as VLAN identifiers and reachability/connection endpoint information from adjacent devices. It is also used to discover the network topology and is finally meant to replace proprietary discovery protocols such as Cisco Discovery Protocol (CDP), Extreme Discovery Protocol (EDP) and others. While its roots are VoIP networks, it can be applied to a broader range of scenarios.

LLDP-capable devices periodically transmit information in messages called Type Length Value (TLV) fields to neighbour devices. This information includes chassis and port identification, system name, system capabilities, system description and other attributes. LLDP-MED builds upon these capabilities by adding media- and IP telephony-specific messages that can be exchanged between the network and endpoints. The new TLV messages will provide detailed information on Power over Ethernet, network policy, media endpoint location for Emergency Call Services and inventory.

The Link Layer Discovery Protocol-Media Endpoint Discovery (LLDP-MED) enhances LLDP with respect of:

- Auto-discovery of LAN policies (such as VLAN, Layer 2 Priority and Diffserv settings) leading to "plug and play" networking.
- Device location discovery to allow creation of location databases and, in the case of VoIP, E911 services. The respective information is compatible to IETF Geopriv WG (civic) and the Emergency Location Identification Number (ELIN) from the VoIP world.
- Extended and automated power management of Power over Ethernet endpoints.
- Inventory management, allowing network administrators to track their network devices, and determine their characteristics (manufacturer, software and hardware versions, serial / asset number).
• It speeds up the start by using the so-called ‘Fast Start’ mechanisms. Proprietary extensions are possible.

<table>
<thead>
<tr>
<th>Discovery MIB</th>
<th>port</th>
<th>device</th>
<th>info</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>Switch</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>C2</td>
<td>IP-Phone</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>D2</td>
<td>IP-Phone</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>F3</td>
<td>IP-PBX</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

Figure 2.2 LLDP (Source: Blatherwick, Romascanu, IETF-68, IETF ECRIT / IEEE Joint Meeting)

The LLDP-MED protocol was formally approved and published as the standard ANSI/TIA-1057 by the Telecommunications Industry Association (TIA) in April 2006.

Known implementations are available from Foundry Networks, Nortel, Extreme, and more.

For HW-VoIP clients supporting LLDP-MED, the solution might be sufficient, for other devices (i.e. PCs) this is questionable. In combination with 802.1X, LLDP-MED can be a SALERNO building block for automatic VPN management, but it is not a full-blown complete solution. Although its message format is flexible and extensible, it is mainly meant to transport Layer-2 related device properties. It also requires a complete LLDP-MED capable network infrastructure.

**Internet Control Message Protocol (ICMP)**

The Internet Control Message Protocol relies on IP and comes in two flavors: ICMP for IPv4 and IPv6. ICMP enables the discovery of all IP-devices in a network by ICMP broadcasting. A common application is ping that makes use of ICMP echo requests and replies. In the scope of IoT@Work, ICMP can be used for single nodes to discover partner nodes. ICMPv6 standard is the basis for auto-configuration functions of IPv6. The protocol is used to both detect the closest gateway that is
responsible of distributing address masks, and also to track presence of neighbouring through a protocol extension called neighbor discovery [75].

2.4.2 Mapping Function Between Link and Network Layers

This function refers to the mapping of an identifier or an address from one layer in the ISO/OSI model to an identifier or address of a higher layer and vice versa. The identifier or the address provides the peering entity at a given OSI layer a unique identity, within the scope of that specific layer. Going bottom-up through the layers, the protocols used to implement this cross-layer function are discussed next.

**Address Resolution Protocol (ARP)**

The ARP [74] is a computer networking protocol for determining a network host's link layer or hardware address when only its Internet Layer (IP) or Network Layer address is known. This function is critical in local area networking as well as for routing internetworking traffic across gateways (routers) based on IP addresses when the next-hop router must be determined.

ARP is most frequently used to translate IPv4 addresses (OSI Layer 3) into Ethernet MAC addresses (OSI Layer 2). This protocol is used to resolve the hardware address of the network interface if there is just the IP address known. In other words, this protocol provides a discovery mechanism for MAC addresses that are associated with IPv4 address. Since Ethernet interfaces need to determine the MAC address of communication partners, an ARP request is send via broadcast if the partner’s address is unknown. Otherwise, for instance Ethernet switches, can query their internal ARP tables to determine a partner’s MAC address.

ARP may also be used as an announcement protocol, called gratuitous ARP, useful for updating ARP tables of other hosts when the sender’s IP or MAC address has changed. Gratuitous ARP may also be used to resolve network problems during start-up or to defend link-local addresses when using the Zero configuration protocol (Zeroconf), a set of techniques that automatically creates a usable IP network without manual intervention or special configuration servers.

RARP, the Reverse ARP Protocol, is used to map Data Link layer addresses to Network layer addresses (RFC903).

**Neighbour Discovery Protocol (NDP)**

The Neighbor Discovery Protocol (NDP) [75] is a protocol in the Internet Protocol Suite used with Internet Protocol Version 6 (IPv6). It operates in the Link Layer of the Internet model (RFC 1122) and is responsible for address auto-configuration of nodes, discovery of other nodes on the link, determining the Link Layer addresses of other nodes, duplicate address detection, finding available routers, address prefix discovery, and maintaining reachability information about the paths to other active neighbor nodes (RFC 4861). The protocol provides a lot of the functions supported in IPv4 by both DHCP and ARP.

**6LoWPAN**

The 6LoWPAN protocol [77], which stands for IPv6 over Low Power wireless Personal Area Networks, provides a series of protocols that interface between IPv6 addresses and IEEE 802.15.4 networks.
2.4.3 TCP/IP Network Information Management: SMI, MIB and SNMP

A network management system (NMS) is a combination of hardware and software used to monitor and administer a network. The architectural model used for the management of information of TCP/IP-based networks is a collection of network management stations, network elements and software components described as follows:

- Network elements or managed devices: Devices such as hosts, gateways, routers, switches, etc. Each managed device executes, at all times, a software component called agent.
- Agents: Software component responsible for performing network management requested by the network management stations.
- Management stations or managers: Execute management applications which monitor and control network elements.
- Network management system applications, which run on the managers.

Management information is viewed as a collection of managed objects, residing in a virtual information store, termed the Management Information Base (MIB). The Internet-standard network management framework is completed by:

- Structure and Identification of Management Information (SMI), which describes how managed objects contained in the MIB are defined;
- Management Information Base for Network Management, which describes the managed objects contained in the MIB [80];
- The Simple Network Management Protocol (SNMP), which defines the protocol used to manage these objects [81].

Formal descriptions of the structure of the objects in a MIB are given using a subset of Abstract Syntax Notation One (ASN.1) [82], as defined by the SMI. In particular, a managed object has a name (object type), a syntax specification (abstract data structure of the type) and an encoding.

SNMP – an Application layer protocol - operations deal with object instances or variables, uniquely identified by their variable names. A variable name is an object identifier (OID) of the form “x.y”, where x is the name of a non-aggregate object type defined in the MIB and y is an object identifier fragment that, in a way specific identifies the desired instance. For example, the one and only instance of the variable sysDescr,iso.org.dod.internet.mgmt.mib.system.sysDescr or 1.3.6.1.2.1.1.1, is identified by 1.3.6.1.2.1.1.1.0.

Top-level MIB OIDs belong to different standard organisations, while lower-level object IDs are allocated by the issuing organisations. The adoption of the OID model permits management across all layers of the OSI reference model, extending into applications such as databases, email, and the Java reference model. This is a result of the MIBs being defined for all such area-specific information and operations.

Initially, the design of the Internet-based network management framework undertook a two-phase strategy. In the short term, SNMP was to be used to manage nodes in the Internet community. In the long term, the use of the OSI network management framework was to be examined. As a result, the initial specifications of SMI (RFC 1065) and MIB (RFC 1066) were developed with the intention of being compatible with the SNMP and OSI frameworks. However, it was soon realised that the requirements of the SNMP and the OSI network management frameworks were more different than anticipated. As such, the requirement for compatibility between SMI/MIB has resulted in development within both frameworks being
suspended. This, in turn, permitted the development of new extensions based on SNMP to respond to new operational needs of the Internet.

2.4.4 Mapping Network Identifiers to Applications

The Real-time Operating System Nucleus Project (TRON): Ubiquitous Code (ucode) and the T-Engine

The Ubiquitous code or ucode offers an end-to-end system that is capable of linking objects with the Internet Of Things. Most of the development work has taken place in Japan and the Far East under the umbrella of the T-Engine Forum and, more specifically, the Ubiquitous Center. The T-Engine is the name for an architecture that is arguably one of the most advanced platforms for ubiquitous computing to be found anywhere in the world. It has evolved from an open computing and communications architecture (TRON project) developed in the 1980's by Professor Ken Sakamura, one of Japan’s leading computer architects.

The basic ucode consists of a 128-bit structure where the first few bits are metadata in the form of a code ID, which then allows the existing code structure to be incorporated and then appended by a serial code if one is not already included in the code structure. For example, the encoding of a EAN-13 product code (the fundamental GS1 product code), is as follows:

- 12-bit code assigned by the Ubiquitous ID Center for EAN-13.
- 52-bit EAN-13 code
- 64-bit serial number, making the complete unique 128-bit ucode unique.

The system has some fundamental differences from some of the other means of linking objects to the Internet of Things. Firstly, in addition to the basic ucode uniquely identifying objects, other ucodes identify space (locations) and even concepts and relationships e.g. name, materials, producer. Thus, to access particular information, it is necessary to use a relational database of the different ucodes.

The Domain Name System and Domain Name Service (DNS)

The Domain Name Service system [78] is designed to provide a mechanism for naming resources in such a way that the names are mappable into IP addresses and are usable in different host, networks, protocol families, internets, and administrative organisations. DNS is used to uniquely name resources connected to a network. It provides a consistent name space independent of resource specific and protocol specific information. The Domain Name Space is a tree structured namespace with nodes containing labels of zero to 63 octets in length. The zero length null label is reserved for the root node of the tree. At each level down the tree, two nodes are managing two logically distinct functions, name servers (in the form of a distributed database) and resolvers. Domain name servers store authoritative data about certain sections of the database (brand or zone of the domain name tree structure) and answer queries about the data. Each internal node in the domain name tree structure requires that the owner of that node provide a domain name server that identifies its children nodes.

Domain resolvers query domain name servers for data on behalf of user processes. Every resource therefore needs a DNS resolver; some resources will also need to run domain name servers. Since no name server has complete information, in general it is necessary to obtain information from more than one name server to resolve a query.
Object Name Service (ONS)

ONS translates an electronic product code (EPC) in an uniform resource locator for RFID technology. Information is stored in a distributed database. If an RFID tag is read, the EPC is passed to a middleware. ONS is used to identify the URL of the EPC where information is stored about the resource. A middleware retrieves the information to provide it in a standardized way. ONS normally holds also information about the product type.

ONS is implemented using DNS with Naming Authority Pointer (NAPTR). A major disadvantage is the missing authorization and authentication connected to ONS queries. Moreover there information that is stored is mainly on class level like the product type.

Finally, ONS can be used to associate EPCs with web services and provides static and dynamic services that can be linked to the EPC.

2.5 Section Conclusions

The above section provides an overview of existing addressing and naming schemes in enterprise and office networks. The IoT@Work project does not intend to invent a new protocol for assigning identifiers and addresses to the different protocol layers. The Internet protocol stack shown in later sections has been so far adapted to the needs of Automation Systems by removing many of the auto-configuration mechanisms – something that the IoT@Work project aims to change. Also, some specific requirements from automation systems need to be met by address management solutions. In this document the latter requirements are investigated by looking at the state of the art in automation systems, as well as the common practice (which is more of a subjective choice of experienced users). Also we define the way an IoT-system approach should modify name and address assignment and management in automation networks. Our Plug and Work approach requires a bootstrapping architecture that integrates both engineering and planning tools in its approach with auto-configuration and flexible mapping of services to device IDs and addresses.
3  Application Layer Service Discovery

In the following section, protocols and technologies are described that allow the discovery of services in the sense of application layer representation of device resources. A resource can be for example a web page, a functionality (like sensing temperature), or server functionality. Here, discovery is considered intentionally in a wide scope so that discovery enabling protocols are also discussed. In a distributed network as described in IoT@Work, nodes need to introduce themselves to other nodes. This discovery process requires publishing and querying services in a secure way. Additionally, nodes must be enabled by a discovery mechanism to share their capabilities and offered services.

3.1 Using Domain Name Service for Service Discovery (DNS - DS)

“DNS Service Discovery is a way of using standard DNS programming interfaces, servers, and packet formats to browse the network for services” (according to [78]).

The main properties of a good service discovery mechanism are given in [79] as the following:

(i) “The ability to query for services of a certain type in a certain logical domain and receive in response a list of named instances (network browsing, or “Service Instance Enumeration”).

(ii) Given a particular named instance, the ability to efficiently resolve that instance name to the required information a client needs to actually use the service, i.e. IP address and port number, at the very least (Service Name Resolution).

(iii) Instance names should be relatively persistent. If a user selects their default printer from a list of available choices today, then tomorrow they should still be able to print on that printer -even if the IP address and/or port number where the service resides have changed -- without the user (or their software) having to repeat the network browsing step a second time.”

The authors of [79] also state that a successful “service discovery protocol should be so simple to implement that virtually any device capable of implementing IP should not have any trouble implementing the service discovery software as well”.

As an example, the own use of DNS for wide area network services is explained. For instance, adding a few records to a DNS server, to automatically advertise selected services to clients with zero configuration on the client side. “When clients get a response packet from the local network’s DHCP server, there’s a domain in that packet, and clients running Mac OS X 10.4 (Tiger) or Bonjour for Windows automatically query that domain for advertised services”. Therefore, with administrative access to the domain in question, it is possible to add the necessary records so the clients will discover web pages, printers, and other network services.

DNS has many features that could still apply to IoT@Work and will be investigated further into more details in the future deliverable D2.2 “A Bootstrapping Architecture”.

3.2 Simple Service Discovery Protocol (SSDP)

SSDP allows the advertisement and discovery of network services without having a dedicated server for that. SSDP is based on HTTP and UDP and serves as a basis of UPnP, for instance. Service announcements are made within the multicast group. Service discovery is made via M-Search method within the group.
3.3 Discovery in P2P networks

Peer-to-Peer – of just P2P – networks constitute an overlay network which applies ITP/IP for communication while providing a highly distributed, serverless architecture with interfaces towards the service or application layer. P2P system can be based on out-of-the-shelf hardware which provides a low-cost infrastructure of nodes or peers, respectively. A sophisticated access and management mechanism guarantees uniformly distributed data hosting on the peers. Jointly with redundancy and self-management features, P2P enables easy scalability, high reliability, and consistency. This holds particularly for structured P2P networks which employ a distributed hash table (DHT) or a centralized index register for address resolution while pure P2P networks develop deficiencies in data discovery.

Consequently, discovery mechanisms in P2P networks vary quite a lot depending on the structure of overlay network.

3.3.1 Flooding

The simplest kinds of P2P networks are the ones with no structure as Gnutella. Such networks do not use any special data storage mechanism: each peer stores its user’s data and hand them over to other peers in case of any request. All a peer needs is a table where is lists other peers it has got in contact with.

Equivalently simple is the discovery mechanism: a peer being asked for particular data delivers the data or, in case no such data is stored at that peer, it asks the peers it knows (and has stored in its table) for the data. While all peer behave in this way the network is flooded and finally the data retrieved is returned to the peer the originated the query. Obviously, this approach lavishes the network with request and is likely to paralyze the network. The only way to reduce the traffic is to reduce the hop count of requests and/or to reduce the number of known peers being queried. A hop count of 5 and a query multicast limited to 100 peers reduces the number of peers eventually queried to $10^{10}$.

In case of an automation system where a lot of network traffic is already passed in the network, this mechanism is not a good choice for IoT@Work purposes.

3.3.2 The DHT Approach

A distributed hash table provides content-addressable access. This means that the index may be any kind of data object which can be converted into a number by using a hash function. This number – the hash value – is used to identify the peer which stores the associated data. Smart P2P protocols bridge the gaps between two succeeding peer IDs. Such Protocols like Chord [10], Kademia [11] or CAN [12] use a maximum hop count for addressing a certain peer of $N$ where $N$ is the binary peer address space width which allows for a maximum of $2^N$ peers. DHTs for secure service discovery exist, e.g. SCAN [25].

The DHT approach guarantees a detection of the retrieved data in case they do exist in the P2P network. If no result is returned by the P2P protocol, no data is available. The disadvantage of the DHTs is the need for an accurate input to the hash function for detecting peers and their associated data. A wildcard search is only feasible with some tricky mechanism methods based on Hilbert’s Curves.

3.3.3 The Napster Approach

This architecture integrates the benefits of both DHT-based P2P and Client/Server. The idea is to centralize the index directory or – in other word – to replace the DHT by a centralized hash table located on a server. Therefore, the operator of the index
server gains full control over the keyword and the peer addresses but also gains responsibility for a uniform distribution of data, for reliability and consistency. A keyword-based request is passed to the index server which replies with the peer address where the source is located. Obviously, the index server is the single point of failure in case of denial of service (DoS) attacks or just simple request torrent.

3.3.4 Prospect

P2P techniques have been mostly used to implement IT and TC applications and solutions. But furthermore, P2P can also be utilized for non-software applications, too, where first results are under development. For instance, the BMBF funded project SemProM (Semantic Product Memory [13]) applies peer-to-peer mechanisms to identify real-world objects (=things!) in logistics and production.

More information on P2P and addressing is presented in [22] and [23].

3.4 Middleware for Service Discovery

The use of middleware as a software implementation connecting different services, components or sub-systems together is often the way to guarantee interoperability independently of the implementation details, protocols, or operating system used by each component. Middleware approaches often list a service discovery, which is often applying some of the previously described protocols. ZeroConf, AutoIP, and UPnP help to establish an initial network configuration, but also to discover services of networked partners. This is the reason why they have to be considered as suggestions for IoT@Work.

3.4.1 Zero Configuration Networking (ZeroConf)

Zero Configuration Networking provides mechanisms to automatically configure and fully establish an IP network. ZeroConf’s address assignment is based on link layer address automatic configuration, for example Automatic Private IP Addressing (APIPA). Its name resolution is based on mDNS or DNS. The service discovery is conducted via DNS service discovery (DNS-SD) [24], [79]. Figure 3.1 presents the evolution of the auto-configuration protocols and service discovery for smarter devices.

3.4.2 Universal Plug and Play (UPnP)

UPnP enables the establishment of services and discovery of network devices for information exchange, especially in (home) office environments. Its implementation is based on SSDP as service discovery mechanism and uses DHCP or AutoIP for address assignment. SOAP over HTTP makes service invocation messages possible. Services are presented by a web server.
3.5 Section Conclusions

This section presented several options for application layer service discovery: DNS, SSDP, P2P, and the middleware-based approaches ZeroConf and UPnP. While DNS itself is a service that provides the translation from (service) names to addresses, SSDP is a Microsoft Protocol by which a service actively multicasts its properties. P2P provides content-addressable access based on keywords by a distributed and replicated index. ZeroConf is a technique which intends to provide a serverless address assignment, name resolution and service discovery. UPnP, finally, is a standard that provides discovery in home office and home entertainment settings. In IoT@Work, we see the potentials of accessing service descriptions and using service discovery to overcome several technical difficulties in commissioning and checking the consistency of the system components with that of the designed project. We therefore foresee the use of both device-level discovery mechanisms and also service level discovery to capture both device Plug&Work triggers, and service Plug&Work as well.
4 Evolution of Directory Services

4.1 Definition and Characteristics of Directories
A directory service identifies resources in a network and provides mechanisms to access these resources, see [8]. Examples for resources are e-mail addresses or network devices.

Directories in the scope of IoT@Work are dynamic. That means that directories may be subject of change and must be updated in a secure and effective way to support plug & work capabilities of the devices in an automation system.

Directories differ from databases in several aspects. First, the focus of directories is on reading rather than on writing information since they are technologically designed for a large read-to-write-ratio. Moreover, information may be stored redundantly in a directory which may be necessary for reliability or performance reasons. For instance, different devices in an automation system store properties of neighboring devices to provide fast access to that information. In this case, replication increases the overall performance by reducing network traffic for the queries sent to a centralized directory server.

Compared to databases, information extensibility is better supported by directories. This refers to flexible data typing, if there is the need to extend existing or to add new types of information. For example, a new device descriptor is introduced to the automation system which comprises new or differing properties. Nevertheless directories support relatively simple transaction models and operations on relatively small amounts of information. Decentralized directories allow distributing information across different parts of the network that supports best the idea of IoT@Work.

Another important question is the scope of decentralization of the directory as there is a trade-off between directory performance and directory managing efforts. In contrast to distributed and replicated directories, centralized directories basically have a lower performance and a longer response time as they provide one single entry point. However, information has to be managed just once while in distributed and replicated directories, data consistency over several nodes requires increased management efforts. On the other hand, distributed and replicated information provision achieves higher performance and lesser response times.

4.2 Applications for Directories in IoT@Work
There are some central applications for directories in IoT@Work. This contains storage, locating, and detecting of:

- configuration steps and guide to indicate how to configure each device (e.g. in a form of script),
- stored pre-defined configuration data of devices,
- discovered services provided by devices or pure data services,
- semantics of services provided by devices,
- associations between devices and services,

In IoT@Work there are two main application areas for directories. They can be used as resource directories that contain the capabilities and properties of resources and devices in order to have an identity management. Since there are many devices with the same hardware and software functionalities but different functions in the
automation system, there is a need for semantic search. A look-up based on keywords supports that search though it may not be sufficient.

Additionally, association directories can be applied to store associations between resources and devices in an automation system. Because of process orientation in many automation systems, a large number of interactions has to be modeled and stored.

4.3 The X.500 Standard

X.500 [14] represents a model for directory services in the OSI world. This model encompasses the namespace and the protocol to query and update it. The applied protocol is called directory access protocol (DAP) and has a very rich data model and manipulation set which makes it “heavyweight”.

A X.500 system is based on a client-/server architecture. Between the client, called directory user agent, and the server, called directory system agent, messages are passed which make use of the DAP. This protocol is used for several operations like binding of directories, disconnecting of directories, reading, writing and deleting of directory entries and searching for properties.

4.4 LDAP (Lightweight Directory Access Protocol)

LDAP is an application layer protocol which is similar to X.500 and can be considered as the lightweight implementation of X.500. LDAP is build to run directly on top of the TCP/IP stack. LDAP provides an information model and a protocol to query and manipulate directories. Like X.500, LDAP is based on a client-/server architecture. Messages are passed between a LDAP-client and a LDAP-server. LDAP directories are spanning a directory information tree. Directory entries are called LDAP objects and are identified by a distinguished name (DN). For further information see [9].

4.5 LDAP/X.500 based implementations for directory services

Here are three implementations presented which are based on LDAP/X.500 standard. The first is Sun Java System Directory Server [15] (Sun Microsystems) that is a LDAP directory server. OpenLDAP [16] is an open-source project of LDAP. Novell’s implementation is called eDirectory [17].

4.6 Similar Technologies

4.6.1 Network Information Service (NIS)

The network information service [18] aims on distributing configuration data between computers in a network. It consists of a client-server directory service protocol that enables a yellow pages service. A NIS maintains a centralized directory of user data for instance.

4.6.2 Universal Description, Discovery and Integration (UDDI)

UDDI [19] is standardized directory service in the context of web services. UDDI nodes are servers which belong to the UDDI registry. A web service provider publishes its service using web service description language (WSDL) and can be accessed by a SOAP message. Moreover, UDDI enables the implementation of white, yellow and green pages.

4.6.3 Directory Service Markup Language (DSML)

The directory service markup language [20] (DSML) enables the acquisition of information from directories via XML based messages.
4.6.4 Meta directories

Meta directories are collecting information and are providing the synchronization and consistency of data from different applications and directory services. These directories act as mediator. DirXML [21] is a mechanism to connect LDAP based directories with each other.

4.7 Section Conclusions

Directories are data management systems which are highly suitable for stable data sets. If there is little write access, directories provide small response times and high availability and reliability. Thus, they appear to be a reasonable tool to store and provide data for an automation system. Today's automation systems rely on the projected setup and configuration data describing the correct state of the system in a centralized engineering station. The engineering station is also misused as network and communication management node. Although this is an implementation issue, the approach in IoT@Work is to clearly examine the way of using network directories in order to separate network and device related management from those needed for the application. The network built-in directories can support configuration tasks related to naming and addressing, and keeping track of service URIs and their corresponding network locations. This approach would hide some of the system complexities from the user, in that networking aspects and some of the configuration steps are delegated to the network and not carried out at the design phase.
5 Secure Device IDs

A Secure Device ID is an identifier associated with a device. The association between an identifier and a device could be a certified binding in the extreme case. A device may have one or more Secure Device IDs and Secure Device IDs may change during the lifetime of a device.

A Secure Device ID and associated metadata should support the authentication of the identity of a device. Secure Device IDs can for example be used in provisioning and authentication protocols to allow a network administrator to establish the trustworthiness of a device and select appropriate policies for transmission and reception of data and control protocols to and from the device.

It should be noted that privacy is not a concern in the IoT@Work project, hence privacy-preserving Secure Device IDs are out of scope.

Secure Device IDs can come in many flavors. They can either be self-generated or assigned by another instance. They can be either certified or not certified. Certification itself can be done in a centralized fashion (e.g. PKI) or in a decentralized fashion (e.g. Web of Trust).

Figure 5.1 shows this classification.

5.1 Self-Generated Uncertified Secure IDs

Self-generated uncertified Secure IDs are often called cryptographically generated identifiers. One example for a protocol using these cryptographically generated identifiers is the well-known HIP protocol (RFC 5201). HIP (Host Identity Protocol) is a host identification technology that separates identifiers of devices from locators (IP addresses). It uses a new set of identities for hosts, replacing IP addresses with host identifiers. The identifiers used are public keys, which are usually self-generated and not certified (HIP also allows certified public keys as identifiers, but this is not often used). The public key is represented by a 128-bit identifier, the Host Identity Tag (HIT). A HIT is generated by hashing the public key. This HIT is used to identify sender and recipient of HIP packets. Consequently, a HIT should be unique as long as it is being used. In the extremely rare case of a single HIT mapping to more than one Host Identity, the Host identifiers (public keys) will make the final difference. For legacy applications, the Local Scope Identifier (LSI) can be used. It is a 32-bit localized representation for a Host Identity, which can for example be used as
address in a socket call. Thus, Local Scope Identifiers act as a bridge for Host
Identities into IPv4-based protocols and APIs.

Identity-based public key cryptography is a paradigm that simplifies key management
and removes the need for public key certificates. For identity, any binary sequence
(e.g. a serial number) can be used that identifies a device in a non-ambiguous way. A
trusted authority (PKG - private key generator) is used that delivers private keys to
devices after having computed the private keys from their identity information (hence
users do not generate those pairs themselves) and a master key only available to a
PKG. In 2001, Boneh and Franklin proposed the first usable identity-based
encryption scheme using bilinear maps (the Weil or Tate pairing) over super singular
elliptic curves to achieve an identity-based encryption method. An identity-based
cryptosystem is made of four algorithms: Setup, Keygen, Encrypt, Decrypt. Setup is
run by a PKG. It outputs a public/private key pair for itself. Keygen is also run by the
PKG. It takes a device’s identity as input and returns the private key for the device.
Encrypt and Decrypt are used for encryption of messages addressed with an identity
and vice versa.

5.2 Self-Generated Certified Secure IDs

Self-assigned, certified secure IDs are often used in subscription models: a requester
generates an identity and the identity is certified by another instance (usually after
some additional checks). For example, a requester could construct a Certificate
Signing Request (CSR), which includes a Common Name and an associated Public
Key as well as algorithm related information. The requester sends the CSR to a
Certificate Authority (CA), which generates a certificate using the information given in
the CSR. In an example use case, the requester generates its ID from some
hardware information (e.g. drive serial number, processor serial number etc.).

5.3 Assigned Certified Secure IDs

Assigned certified secure IDs for devices are often contained in so-called device
certificates. A device certificate is a certificate issued to a specific device, like other
ID certificates, it allows to bind an identity to an associated key. The identifier given in
the certificate is usually assigned by a central instance, e.g. the manufacturer of the
device or the administrator of the network the device should have access to. Such
certificates can be used to associate a device identifier with a public key. The
public/private key pair can then be used to authenticate devices. Device
authentication is e.g. part of IEEE 802.1X standard for port-based Network Access
Control. IEEE 802.1X can use device certificates (amongst other credentials) for
network access decisions.

Presently there is no standard identifier for IEEE 802 devices that is cryptographically
bound to that device, nor is there a standard mechanism to authenticate a device’s
identity. The goal is the authentication of entities attached to a network in a secure
fashion; e.g., by means of the mechanisms defined in IEEE 802.1X. Moreover, a
standardized device identity facilitates interoperable secure device authentication.

According to IEEE 802.1ar a secure device identifier (DevID) is a cryptographic
credential in form of an X.509 certificate bound to a particular device. It supports
authentication of the device’s identity. DevIDs are used to establish and authenticate
the identity of the devices that contain them. Initially, devices contain an IDevID
(Initial DevID) credential installed during manufacturing. Subsequently, devices can
install locally significant LDevIDs (Locally significant DevID). Locally significant
identities can be securely associated with an initial manufacturer provisioned DevID,
and used in provisioning and authentication protocols to allow a network
administrator to establish the trustworthiness of a device and select appropriate
policies for transmission and reception of data and control protocols to and from the device.

IEEE 802.1ar provides:

- Globally unique manufacturer provided Initial Device Identifier (IDevID),
- Locally Significant Device Identifiers (LDevIDs), LDevID is bound to the IDevID in a way that makes it impossible (to within a known and exceedingly small bound) for it to be forged or transferred to a device with a different IDevID without knowledge of the private key used to effect the cryptographic binding.
- Options provided based on X.509 specifications
- Usage models for network-centric enterprise scenario and home network devices
- Definition of key attributes required for device identity, security requirements, owner, issuer, replication etc.

5.4 Assigned Uncertified Secure IDs

An assigned uncertified secure ID can for example be a simple bit-string identifying a device (e.g. a hostname) together with an associated symmetric key or password for authentication. Bit-string and associated credential (symmetric key or password) are assigned by a central instance. This instance has a list of bit-strings and associated credentials (symmetric key or password) used for authentication. Kerberos is an example for a security system using assigned uncertified secure IDs (but not for device identification but for user authentication).

5.5 Section Conclusions

Secure device identifiers will become the basis for various services for the more open future automation environments. Among these services will be authentication, authorization, auto configuration, localization support, etc.

The intention for IoT@Work is not to invent completely new device IDs but instead to apply established and standardized identifiers and enhance them with features necessary for open automation environments.

Some of the most relevant features for the IoT@Work environment are binding of the ID to the hardware, support of multiple IDs, binding of additional meta data to the IDs, ability to check the IDs beyond domain borders by using standardized security protocols. We see that these requirements could be best fulfilled by IDs falling in the categories of “Self-generated Certified Secure IDs” or “Assigned Certified Secure IDs”. Therefore further investigation will concentrate on possible realizations and integration of Device IDs out of these categories.
6 Industrial Configuration Tools

6.1 Industrial Engineering Tools and Adopted Protocols

Industrial automation systems need tools for consistent and plant-wide configuration of distributed control systems (DCS), and field-buses and devices that could integrate sensors and actuators of different vendors in any control or engineering system. On the one hand offline device descriptions (DDs) for device management are needed and usually provided by the device vendor. On the other hand configuration tools and start-up protocols are required. Some related technologies are described in the following section.

6.1.1 Common Engineering Process of an Industrial Plant

An industrial plant has to be initialized and configured before starting the actual manufacturing process. The first challenging part of a successful integration is the adaptation of engineering processes. As an example the basic procedure of the engineering aspect for an industrial network is shown in Figure 6.1. As a starting point, it is necessary to have sufficient device description (DD) information for the entire industrial network.

1. The offline device information represented either as a generic station description mark-up language (GSDML) file or an electronic device description (EDD) file has to be made available to the engineering tool (Engineering Station).
2. The engineering tool (Engineering Station) creates a project with configuration and parameterization.
3. This information will be passed to the controller (PLC).
4. The controller configures all devices in the start-up phase of the system.

![Figure 6.1: General Workflow of Engineering an Industrial Process](image)

(1) GSD import and net configuration in the Engineering Tool  
(2) Device configuration and download in the PLC  
(3) Automatic cyclic process data exchange between PLC and decentralize field devices  
(a) EDD import and device configuration in the Engineering Tool (EDD host)  
(b) Acyclic data exchange between Controller and decentral field devices for device configuration and asset management

Whenever a new device is added, the whole procedure has to be repeated, starting with step 1 [37].
6.1.2 Device Description

The Electronic Device Description Language (EDDL) is an international standard and used by major manufacturers to describe the information that is accessible in digital devices. Electronic device descriptions are available for over 15 million devices that are currently installed in the process industry. The technology is used by major process control systems and maintenance tool suppliers to support device diagnostics and calibration [31]. Given suitable tools, the object dictionary of a device can be configured by editing an electronic data sheet (EDS) file and uploading the variable values to the device. An example of EDS is the XML based GSDML (Generic Station Description Markup Language) file that describes the expected implementation of an IO device. It includes elements like device identity, device function, application process, etc. For I/O data, the GSDML file describes the structure of the cyclic input and output data transferred between the programmable logic controller (PLC) and the IO device. Any mismatch between the size or structure of the input and output data and the actual internal device structure generates an alarm to the controller [32].

6.1.3 Generic Interfaces for Device Management

Device description files (e.g. HART, FDCML, GSD, GSDML) are normally used by vendor specific engineering tools to configure a certain system. A more generic approach is to use generic interfaces, such as the field device tool (FDT) or tool calling interface (TCI) for the device management which allow an easy integration into existing engineering tools. Finally, the field device integration (FDI) standard, which combines FDT and EDDL, is also an important framework.

FDT is a suite of technologies defining a generic communication interface between field devices and the engineering tool. The device type manager (DTM) represents the actual field devices and is part of FDT. Basically, a DTM is a software module (a kind of driver) that comes with each device. DTM encapsulates the device's configuration, functions, parameters and describes the user interface. Communication Drivers that represent the communication hardware are needed for connecting the field device to the automation software. They are referred to as “Communication or Gateway DTMs”. All device manufacturers offer device DTMs (or complete libraries of them) for their equipment. For the missing pieces, several solutions exist, such as profile DTMs for PROFINET devices, generic DTMs for HART devices (supporting universal commands) and DTMs that are compiled from device descriptions.

Another important standard is TCI, developed by Siemens specifically for the PNO, to access the complex device tools of engineering system. The aim of TCI is also to ease connecting various proprietary device tools with the engineering system. Currently TCI is included in the Simatic Manager from Siemens and specified for the field bus Profinet DP and Profinet IO. However, it could be extended to other field buses [33], [34] as well.

FDI is an initiative driven from various interest groups such as the FDT group, the HART communication foundation, the OPC foundation, the Profinet User Organization, and some others. It aims at being a robust and future proof technology for device integration. This is achieved by incorporating the best aspects of EDDL and FDT and eliminating the existing redundancies. The developed FDI concept integrates EDDL advantages, like platform independence, ease of use, and robustness, with the FDT benefits, such as unlimited functionality, extensibility, and market differentiation, into a single architecture [35].
6.1.4 Existing Engineering Tools

An engineering system supports the user through all the stages of a development process for automation solutions, which are:

- Installation and management of projects
- Configuring and parameter assignment of hardware and communications
- Program generation for target systems
- Loading programs on target systems
- Debugging a project
- Testing the automation plant

An example of an engineering tool is Siemens’s SIMATIC STEP7, which is designed for professional use with SIMATIC controllers. It includes powerful tools and functions for various tasks involved in an automation project.

PCWorX was developed by Phoenix Contact and is a further example of an engineering software. It is designed for programming of IEC 61131 based PLCs and the configuration of communication networks like Profinet and Interbus. It is suitable for use in distributed automation systems as well as for conventional centralized automation.

Freelance is the engineering solution of ABB and combines tools for configuration, programming and visualization of machines and automation systems.

All tools mentioned above provide support for IEC 61131 based programming languages and allow interpretability using OPC (Object Linking and Embedding (OLE) for Process Control) for vertical integration.

6.1.5 Automation Application Programming Languages (IEC 61131)

IEC 61131-3 is a world wide standard. It harmonizes the way people look into industrial control by standardizing the programming interface. This includes the definition of the language sequential function charts (SFC), used to structure the internal organization of a program, and four interoperable languages: instruction list (IL), ladder diagram (LD), function block diagram (FBD) and structured text (ST). Each program is additionally structured by means of modularization and declaration of variables. This leads to an increased re-usability, a reduction of errors and a higher efficiency. In addition, IEC 61131-3 structures the configuration of control systems [IEC 61131-3].

6.1.6 Startup Phase of an Industrial Controller and Corresponding Communication Protocols

An industrial network (for example Profinet IO) consists of an IO controller and one or more IO devices. The connection of the IO controller to the IO devices is called an application relationship (AR).

The IO controller sets up an AR to every configured IO device in the project. This is realized by a UDP/IP transmission with Distributed Computing Environment / Remote Procedure Calls (DCE RPC). The possible IP addresses are defined in the planning phase of the IO controller within the engineering tool. The IO controller has the responsibility to assign IP addresses to all different IO devices. The identification of the IO devices is not performed by the use of IP addresses, but rather with their Profinet names. During the engineering phase the engineering tool assigns a name to every IO device using the DCP protocol (Discovery and Basic Configuration) as
shown in Figure 6.2. This name is stored in a non volatile memory in the IO device and must be unique for the whole network.

![Diagram of IO controller and IO device with steps for address assignment](image)

**Figure 6.2: Assignment of the IP addresses to the IO-Device, normal sequence [36]**

When the IO controller starts processing, it first verifies that all IO devices are reachable with their names using a DCP-Identify request. If an IO device has not yet had its IP address assigned, the IO controller verifies with an ARP-Request, that the name is not already in use and then assigns a valid IP address to the IO device using a DCP-Set. The IO controller then sends the AR-Setup data over the assigned IP address. Inside the AR the IO controller specifies a communication relationship (CR) for the cyclic IO-data to be exchanged between the IO controller and the IO device. Once the initialization sequence is complete (cf. Figure 6.3) the IO device sends its input data in the specified cycle time to the IO controller and the IO controller sends the output data to the IO device over a Profinet RT-Frame with Ethernet Type = 0x8892. Thus, for cyclic IO-Data, the IP address is not used anymore, only MAC addresses are used [35].

![Diagram of Profinet AR setup sequence](image)

**Figure 6.3: Profinet AR setup sequence [36]**
6.1.7 Section Conclusions

Due to the described mechanisms in the previous subsections and the tight coupling of the network to the application, the current engineering process of industrial automation systems is rather static and does not support agile manufacturing systems as addressed by IoT@Work.

Flexibility basically means adaptivity to changing operating conditions. An adaptable automation plant allows to add new components or remove existing ones. For creating an adaptive manufacturing process, an engineering system should keep track of dynamic changes and adapt the application accordingly. This kind of adaptivity is still an open topic and requires further analysis in terms of how various components influence the whole system behaviour. Further, the definition of interfaces and criteria to ensure adaptability in perfect harmonization is needed.

The IoT@Work architecture will be able to decouple the automation application programming from the underlying network operation by means of additional IoT@Work specific services and newly introduced abstraction layers.
7 Scenario Driven Requirements on Naming and Addressing

7.1 Introduction to the Requirement Methodology

Besides the enablement of self-organization and auto-configuration during the commissioning of manufacturing systems, there are other system generic requirements that are driven by situations and use-cases studied and detailed in Deliverable D1.1. The resulting three scenario clusters are namely Agile Manufacturing, Large Scale Manufacturing, and Remote Maintenance. These three scenarios are used to verify the importance of the IoT@Work initial hypotheses and the main objectives in an early step. The scenario also proved a powerful method to identify the system model that needs to be taken into account during the drafting of architecture requirements and then designing the functional blocks of the latter architecture. In this section the impact of the scenarios on the naming and addressing mechanisms adopted in the manufacturing world are analyzed. The result of this section is an initial list of requirements for the naming and addressing architecture components. This analysis would then fit into the functional blocks of a future bootstrapping architecture.

7.2 Large Scale Manufacturing and Agile Manufacturing Scenarios

The agile manufacturing scenario details the reasons for agility and the way this is defined in IoT@Work context (see D1.1, Section 6). The large scale manufacturing scenario is a generalization of the agility to a more complex manufacturing environment like that of FIAT, whose manufacturing sites include a number of factories.

7.2.1 Cloneable Subsystems

A smaller or larger part of the installation will be installed multiple times in the factory (i.e. many identical welding cells). It should be easy to copy the configuration while names and addresses are automatically generated. This requirement applies mostly to large scale manufacturing sites. Whereas the focus of IoT@Work at the moment is on dealing with naming and addressing issues in general in around machines and cells, the problem of automating this process for large systems can lead to a different bootstrapping architecture. This can be compared with providing a DHCP solution for a home networking setup where a home user normally ends up with a self-configuring subnet. A large campus or enterprise network might require a different DHCP solution and NAT-based subnets that need to interoperate with each other. The solution provided for a home network cannot immediately scale to the enterprise system if kept unchanged.

7.2.2 Extensible Subsystems

It should be easily possible to add new services, new technologies, sub systems or devices. This will e.g. affect the number of IP addresses needed for the subnetwork.

7.2.3 Company Wide Access

Devices and services should be reachable from anywhere in the company. This requirement mainly addresses issues considering address rooms (NAT or flat IP address space) and probably structuring of security realms.
7.3 Remote Access Manufacturing

7.3.1 Directory entries and data-base management

The directories listing device IDs at remote maintenance provider should match those of the factory owner.

The IERC Strategic research roadmap suggests developing solutions for managing, at the same time, different "names" for the same object. These names can, for example, identify the object according to a "naming strategy" of the provider (e.g. "Robot NG3-SN xxyyzz") or of the owner (e.g. "Mirafiori Robot NG3 Line 1-Cell 3"). Objects can also be identified as members of a specific class (e.g. "Robot NG3") especially if we take into account the IoT@Work objective to dynamically plug new devices at the network level. We must be able to manage different naming approaches and to manage "links" between different names of the same object. The remote provider will have its object's name (in its Name System Management service) pointing to the corresponding object's name within the factory owner Name System Management Service. If the owner updates its internal ID, this will not affect any external links, provided that the "internal system" maintains links able to redirect the name resolution from the "old name" to the "new one". This implies also that "names" cannot be removed from the system unless the referred object has been removed from the system too.

7.3.2 Dealing with Address Changes

If a locator (usually IP address) changes at the factory owner, this change has to be kept transparent for the maintenance provider.

This is requirement should be an improvement of current state of the art. Often, maintenance providers have to maintain a common list of network details of their supplied equipment, even if the network is owned by the factory owner. For the maintenance provider, having a "non routable" IP address outside the private domain (i.e. the factory plant) doesn't provide any useful information. The intervention of a network administrator to set-up a connection for the external maintenance provider is often the only way to connect to the right device. Identifying the correct device is done through comparing data bank entries of both administrators with the pre-configured IP address.

7.3.3 Remote Access

Connections/Updates/Access requests should be routed automatically to the right device/service locator.

As common state of the art practice, remote connections to a device is guaranteed through two different approaches:

- Connecting to the device through a GPRS gateway to the public Internet (i.e., using mobile network infrastructure). These connections do not use access the factory’s network nor require approval of the factory owner. The factory owner might specify the type of read/write access rights and ownership of the data being accessed.

- Connecting to a device through the factory owner’s network, with limited and controlled access. These connections are short lived and are set-up through firewalls of the factory’s networks. These connections are routed to reach the end-device through the locator of the device or service being remotely accessed.

The above stated requirement applies mainly to the second scenario, where remote connections are enabled through the factory owner’s network. For example, when setting up a remote connection to a specific robot within Cell xx in Linee yy at the
Mirafiore plant, the maintenance provider should use only the known ID they manage in their directories for that specific robot. The IoT@Work “naming system” has to resolve on both maintenance provider side and the Mirafiore plant, the addresses of the gateways used to provide a secure connection over the public Internet. The “naming system” has also to resolve the IDs of the specific robot into an internal one. Updates, to the IDs on either side should not affect each other. There needs to be some intermediate system (indirection infrastructure) within the factory’s infrastructure (or plant) so that the plant owner policy could be applied (e.g. checking integrity, virus scan, specific test suites ...etc.).

7.3.4 Cross-Checking Device Identity

Semantic information of a device can be used to check the identity of the device remotely.

The use of directories alone with lists of locators of each device and service can be in some cases insufficient. An example of mixed identities happens when wiring devices of the same nature (e.g., I/O devices). A common mistake is when a wire is plugged into the wrong I/O device leading to an error that discovered during testing of the newly commissioned system. To avoid this, semantic information related to the location: geographic, topological, and hierarchical (line ID, Plant, Robot arm ID, etc), should be stored next to the locator of the device to allow an automatic identity check. When setting up a remote connection to a device in the factory infrastructure, this extended directory information can be cross-checked automatically before setting up a remote connection.
8 Future Trends and Needs

8.1 Standardization Activities
Throughout the years, there have been efforts to standardize several aspects of automation technology. In the following subsections, a few recent standards, that are relevant to communication, networking and security, are briefly discussed.

8.1.1 AutomationML
AutomationML stands for “Automation Markup Language”. It is an effort to develop a neutral data format that is an open standard based on XML technology. It defines standardized mechanisms for the storage and exchange of plant engineering information. Its main goal is to integrate the heterogeneous engineering tools used nowadays in Factory and Process Automation. Such tools are used in different disciplines such as: motor management, visualization (HMI), mechanical engineering, PLC-based control (discrete, process, motion), robot control and DCS-based process control.

The basic architecture of AutomationML consists of the standards CAEX [38], PLCopen XML [39] and COLLADA [40]. CAEX acts as the top level format and stores the plant topology, COLLADA stores geometric and kinematic information, while PLCopen XML is used for the storage of sequences and behavior [41].

AutomationML supports the storage of the following basic concepts [41]:

Geometry – A 3D description of a data object. This information is stored as separate XML files. AutomationML defines special reference mechanisms to link CAEX data objects with a COLLADA file. With this information, the complete geometry scene can be calculated automatically.

Plant Topology – This is the description of a plant as a hierarchical structure of plant objects and is the top level format within AutomationML. Each object may have individual properties and interfaces. For plant topology storage CAEX is used, therefore becoming the high level integration framework.

Kinematic – It describes the physical interconnection of 3D objects and their geometric dependencies for the motion planning. This information is also stored by means of the data format COLLADA as separate XML files.

Behavior description – An object’s behavior is described as a sequence of actions. These are stored by means of the data format PLCopen XML as Sequential Function Charts (SFC) including the I/O mapping to logical and physical variables.

References and relations - References describe associations from CAEX objects to externally stored files. Relations specify interrelations between CAEX objects. These are additionally used in order to link information in the top level format that is stored externally.

8.1.2 ISA99 - Industrial Automation and Control System Security
This standard proposes procedures for implementing electronically secure manufacturing and control systems and practices for assessing electronic security performance. The recommendations are directed towards an audience that includes: users, system integrators, security practitioners, and control systems manufacturers and vendors.

The ISA 99 standard comprises two parts [41]:

References

The concept of manufacturing and control systems electronic security encompasses all types of plants, facilities, and systems in all types of industries. From this point of view, manufacturing systems include:

- Hardware and software systems such as DCS, PLC, SCADA, sensor networks, and monitoring and diagnostic systems.
- Associated internal, human, network, or machine interfaces used to provide control, safety, and manufacturing operations functionality to continuous, batch, discrete, motion and other processes.

Of particular interest is the technical report ANSI/ISA-TR99.00.01-2007 “Security Technologies for Industrial Automation and Control Systems” which provides a current assessment of various cyber security tools, mitigation counter-measures, and technologies that may effectively apply to PLC or DCS based control systems regulating and monitoring numerous industries and critical infrastructures.

8.1.3 ISA 100 standards – Family of wireless standards

ISA-100 is a set of standards whose goal is to enable a single, integrated wireless infrastructure platform for manufacturing sites. The family of standards that defines “Wireless systems for industrial automation and control applications” encompasses different systems for different applications which have unique requirements. Such is the case of field level control, device level monitoring, and backhaul/backbone communication. The ISA 100 standard considers coexistence among different control and non-control communication systems and aims at providing mechanisms to mitigate interference between them.

Due to the open, easily accessible and shared medium, the use of wireless technologies in factory and process automation poses significant challenges within the context of security. For this reason ISA 100 defines security and management schemes for wireless devices serving applications that range from monitoring to non-critical control. Part of the approach involves choosing radio bands and security techniques that are deployable throughout the world. Because security is a major design aspect of ISA100.11, it considers total life cycle such as configuration, operation, maintenance, etc. Moreover, security is considered throughout the whole system and not just at the PHY layer or MAC sub-layer. The standards leverage security aspects existing technologies such as IEEE 802.15.4-2006 standard, which allows for reduced costs, quicker implementations, and a broad consensus of security experts [43].

8.1.4 Other related standards

Two other related standards are:

- IEC 61499. Defines an open architecture for the next generation of distributed control and automation.
- ISO 10303. It is a standard for the computer-interpretable representation and exchange of product manufacturing information.

However, these standards do not define actual communication aspects or mechanisms.
8.2 Automatic configuration and self-organization

8.2.1 Definition and characteristics of automatic configuration and self-organization

Self-organization (also referred as self-management in the scope of self-* capabilities) has already been discussed for a long period (since the late 1960s) in science and points to a specific control paradigm for complex systems. The most prominent example is the small-world phenomenon that Stanley Milgram researched in 1967 [44]. The fact that every peer can be reached by a small number of hops and peers tend to be clustering according to their similarity has big implications to self-organizing networks. Originally, the term of self-organizing systems emerged from biological systems and is based on algorithms organizing autonomous components [45]. Most solutions in the scope of self-organization make use of methodologies similar to biological systems [46].

But nevertheless, there is no common definition of self-organization in science today. The term self-organization is context dependent. This is due to the fact that self-organization describes phenomena of specific applications and processes. This section tries therefore to apply most commonly used definitions in order to solve problems in industrial automation systems that can learn from self-organization approaches. A self-organizing system can be understood as a set of nodes interconnected via links in a dynamic and adaptive network [47].

Self-organization is a process in which structure and functionality (pattern) at the global level of a system emerge solely from numerous interactions among the lower-level components of a system. The term “self” refers to the missing external or centralized control. In consequence, self-organization can be seen as the interactions (communication) between nodes in the network leading to globally visible effects, e.g. the transport of messages from a source node to a sink node. The system’s components interact in a local context with local and necessarily distributed control either by means of direct communication or environmental observations without reference to the global pattern in a peer-to-peer fashion. This states that self-organizing systems are massively distributed and have no globally valid state information. This leads to spontaneous interaction of in general heterogeneous units in a local decision making process [48].

Transferred to the scope of IoT@Work project, self-organization will be interpreted as the capability of an automation system to providing help to automatically integrate, configure and administrate automation devices in a plug & work manner to achieve adaptability to environmental changes. In this context, the term automatic configuration of systems and devices is regarded as a subset of self-organization.

There exist a lot of definitions and interpretations of self-organizing systems.

In this scope, the following characteristics and properties are identified to describe self-organization [49] and [50]:

- Emerging structures: This describes the fact that visible effects and structures emerge from the interaction in the system and are not based just on its parts.
- Energetic openness: The system has the ability to evolve by obtaining input from the environment.
- No centralized control: The subsystems act autonomously without being controlled by other subsystems or control units.
• Complexity: The overall system behavior may be unpredictable and complex based on interaction of autonomous subsystems.

Scalability: There are no performance limitations arising from adding new subsystems or new units to the organization.

Moreover self-organization briefly refers to redundancy, autonomy and self-reference. By definition, self-organization touches communication issues, so that network structures, topologies and routing of information have to be considered. Coordination, collaboration, and task allocation are directly linked to communicating partners. In order to work in a self-organizing way, a system’s components need semantic information about communication peers and their roles in the network. These aspects emphasize that the communication network is the backbone of self-organizing systems.

Self-organized systems can be built upon centralized control (e.g., use of a single server), but there is one major disadvantage regarding scalability and adaptability in this. However, scalability can be seen as one important requirement to realize self-organization in automation systems. So, distributed systems are much more appropriate to foster self-organization in a system because they provide decentralized self-organizing algorithms. This enables rapidly adapting services in a highly dynamic environment, what especially gains importance in plug & work applications. This is the reason why the following aspects focus on distributed systems and their enabling technologies and have a scientifically based methods (e.g., graph theory) for modelling purposes [51].

8.2.2 Self-organization in automation systems

To link to the mentioned aspects above, self-organization should reduce the effort linked to configuration of a system and to facilitate network operation. Especially the ever rising complexity in automation systems makes the domain of manufacturing having a look for principles how to scope with this challenge. There are new paradigms in manufacturing and automation discussed such as reconfigurable and adaptive manufacturing systems that might give an answer to it. “A reconfigurable manufacturing system is designed for rapid adjustment of production capacity and functionality, in response to new circumstances, by rearrangement or change of its components” [52].

Self-organization is therefore an inherent property of those systems and automation systems might profit from the progress made in the field of IoT and self-organization. Another big issue is the costly engineering of automation systems, especially regarding process control systems and their integration in manufacturing execution systems [53]. The discussed topics of plug & work mechanisms and service discovery in IoT@Work will make a contribution to achieve smaller engineering costs and efforts.

8.2.3 Requirements in automation systems

The huge challenge in automation systems to realize self-organizing properties comes from the fact that every device has a specialized and dedicated role on the field or on the control level. These roles, for instance for I/O-devices or PLCs, have to be defined in order to have a bases for an organizing system. In this context, semantics are a good way to describe, represent service-oriented features on top of a control task in an IoT@Work middleware.

In IoT focused automation systems, every device must obtain a unique IP address. Regarding auto configuration in an automation system, one can learn from mobile ad
hoc networks where routing and configuration of IP addresses is based on network dynamics. This implicates that a distributed approach is best fitting. During the last few years, there is a continuing trend towards distributed control systems in industrial automation [54] and [55].

The many advantages of distributed control systems regarding flexibility and adaptability come with some challenges that have to be met. Distribution of control leads to a higher complexity caused by communication needs and heterogeneous resources. If relying on a pure distributed system, the storage of configuration data is not centralized any more for instance. Mechanisms must ensure that the automation system does not behave unpredictable [56].

The subset of reconfiguration, e. g. adding new devices like sensors or actors should work in a plug & work manner, has high potential to reduce engineering costs of an automation system. This means that newly integrated devices in an automation system should work without human assistance [54], [62].

As long as self-organization is seen on the network level or on top of the control level in an automation system, there should be no need for real-time requirements. But every interaction via interfaces with the control level has to be checked if real-time issues are affected. Moreover, self-organization comes with properties like energetic openness and emerging structures so that security issues are highly affected, especially as far as automation systems are concerned.

Automatic configuration of automation systems should comprise at least the following aspects: the definition of the role of an automation component and its discovery in a distributed IoT network.

8.2.4 Role definition

The role definition is conducted during the engineering steps in an automation system. This definition is an important basis for each device to know its task and properties and to be capable to provide services to other devices. The introduction of roles in an automation system would be a major step towards the abstraction of engineering information from the hardware. At the moment, there is a strong link between hardware and the application side which leads to disadvantages like inflexibility and monolithic architectures. Roles would encapsulate the functionality of devices with standard interfaces so that modularity and flexible adaption could be achieved. The introduction of roles is especially important when it comes to plugging new devices to the network. A keyword identification of devices is not sufficient in heterogeneous environment like automation systems where a device can capture many different roles.

There are two ways how a role can be assigned to a device: the device is preconfigured and announces its capabilities in the network by being plugged in or a device has to retrieve its configuration from the network. Nevertheless, a basic role definition has to be already implemented on the device.

8.2.5 Service discovery for self-organization

The following points comprehend mechanisms of service discovery. These mechanisms assume that information is distributed in the system as IoT@Work implicitly states.

Referring to implicit coordination, nodes can use past data from previous searches to discover resources. This mechanism is applied in NeuroGrid. The grid conducts distributed searches using semantic routing while each peer node is associated with a keyword [56].
[57] presents the storage of information about associations and interactions between peers according to social networks and social interactions between humans.

In an ant-based approach, peers are continuously recording the queries that are passing the peer and are recording which queries neighboring peers are able to answer. Based on this information, a peer can route queries itself to the peer that is most likely capable to answer requests [58].

[59] proposes keyword similarity and small-world clustering to support service discovery.

As a conclusion, it can be stated that because of the distribution of information and the requirement of self-organization, peer-to-peer networks and algorithms should be considered as the enabling technology for service discovery in a self-organizing system.

8.2.6 Methods and techniques of self-organization

Basic methods and models to create self-organizing systems are described in. Positive feedback is a further important mechanism that enables self-organization. Amplification in positive feedback loops leads to the emergence of patterns. Such patterns can be used as an observable system state.

There is a lot of tool support for modelling self-organizing systems. For instance, the graph theory or special rules allow modelling dynamics in a system [51].

Plug & work is strongly interconnected with the slow growth of a network and addresses the (re-) wiring of communication links as described in [61]. An important aspect in self-organizing automation systems is direct communication to neighboring components what leads directly to a peer-to-peer approach known from computer science.

**IPv6 automatic configuration mechanisms**

Stateless auto-configuration

With this mechanism, there is no requirement for manual IP configuration of hosts. A host can generate its own IP address by using local and non-local information. The router provides non-local information, for instance the subnet. The host provides a unique local identifier. Combining these two elements, an IP address is generated.

Stateful auto-configuration

In contrast to a stateless auto configuration, there exists a so-called stateful auto-configuration. In this context, configuration data is provided by a server.

8.2.7 Design paradigms of self-organization

The following design paradigms of self-organization for communication networks are taken from [46]. In case of communication, automation systems rely on network communication mechanisms, so that these design principles can be adopted.

**Designing local behavior to achieve global properties**

One possibility of achieving global system properties through local behavior is clustering. The goal is to cluster a network. Each node belongs to a cluster and has n-hops distant cluster head. The cluster head is in charge of the inter cluster communication. With regards to automation systems, a cluster can be a sub system
like manufacturing facility. There are many facilities in a plant that have a lot of technological dependencies.

**Tolerating imperfect coordination by implicit coordination**

Conflicts can be basically tolerated in a self-organizing system if they can be managed locally, detected easily and are restricted in time.

Moreover implicit coordination can be applied in the case of self-organization. This means that information passing is not based on message exchange but rather by detecting what is going on in a node’s vicinity. A node passively listens to the network traffic that passes and retrieves information about its environment without having direct message exchange to other nodes. Assuming that an automation system is already set up, only few changes have to be done (e.g. device exchange). So, automation devices could easily listen to network traffic if they have appropriate network interfaces and capabilities.

**Minimizing long-lived state information**

In order to avoid synchronization, the system should not rely on long-lived information. One way to achieve this is the application of discovery mechanisms. Discovery ensures that information does not have to be centrally stored.

**8.2.8 Protocols for adaptability to changes**

As there is no central control monitoring the environment for changes, each node has to organize knowledge and information updates itself. This goes along with the IoT approach that each automation device should have its memory to save environmental data and is in charge of making own updates if there are changes.

**Architectural considerations**

Self-management software must support the automatic configuration of the interaction of the single components. The objective is to minimize the necessary configuration effort by humans. So, there is a three-layer reference model presented in [62]. The basic layer is the component layer where sensors and actors are located and interact with each other. Changes are reported to a middle layer, the change management layer. This layer executes actions when changes are detected. The most upper layer is the goal management layer that is in charge of overall planning.

**8.2.9 Limitations of self-organization**

First of all, self-organizing systems have reduced controllability because there is no global state information retrievable. Information is basically distributed in the system and this leads to an unavailable or limited determinism. There must be a fair balance between determinism on the one hand and self-organizing capabilities on the other hand in a concrete automation application. System tests and validating the system is complicated and complex because the way of obtaining results and states might be different from the previous ones.

One of the most challenging tasks to cope with is linked to security issues. One of the inherent characteristics of self-organization is openness. This property is a major concern when securing systems. Therefore, the design of self-organizing systems must be accompanied by a security analysis, especially in the field of industrial automation.
8.2.10 Future Challenges

In this section, two central aspects of self-organization in automation systems with regards to IoT@Work will be discussed.

**Plug & work in automation systems**

Plug & work mechanisms in automation systems are a major challenge in terms of self-organization. There is a strong need for engineering support because of increasing complexity of those systems. This functionality requires a self-describing capability of the device and the possibility to publish these properties in the network [63].

One has to distinguish between the engineering phase and the run-time period of an automation system. Engineering will always require human intervention and support to define the automation task of the system. During the run-time period, an automation system just needs little change. But the important aspect is that most setup and configuration information can be implicitly or explicitly retrieved from the system. In this case, self-organizing plug & work features can give a lot of support to handle complexity.

In the engineering step of an automation system, self-organizing features like plug & work can just give basic support to humans. Plug & work mechanisms could for instance contain supported setup of communication links between devices and reduce engineering costs in this way.

A first approach has already been published describing a self-configuring component as a first step towards plug & work. In [64], the authors present a service-oriented approach for automation system which incorporate components with self-configuration functions. The whole live cycle is subdivided into two sections of which the first one covers the following steps:

1. Initial component configuration
2. Component start-up as a reaction on the initial configuration and for commissioning the configuration
3. Device failure
4. Device replacement if necessary

These steps proceed in the traditional way, particular an administrator has to be involved if a hardware replacement is needed. However, the following steps are now automated such that the administrator will not have to elaborate and incorporate configuration data manually.

- Device identity recognition
- Configuration data retrieval
- Configuration and start-up

For the two steps, services are provided. The device identity recognition may be accomplished (beside a manual adjustment) by a local label like an RFID tag which is read by the new component. Alternatively, the communication environment could provide a fixed address which is taken as the identity and, finally, other components in the network will be able to provide the identity according to the history recording.

Once its identity is known, the component is able to download its specific configuration data from a certain device hosting these data. The address of that device is calculated by using the identity data as input for an overall explicit address calculation function. From the accordingly detected host, the component downloads
the configuration data and installs them appropriately. Then, it reboots to make the configuration work.

Identity recognition, configuration data download and installation and eventually rebooting form a sequence of steps being equivalent for all components. After rebooting, the components behave in their individual way.

Plug & work mechanisms become more and more important while the complexity of automation systems is increasing. There are two technologies coming from the office world that can serve as basis to realize plug & work. Zero Configuration Networking is currently used for configuring network printers [65]. Universal Plug and Play (UPnP) is used as configuration mechanism in home audio and video applications [66].

Automatic configuration in automation systems always affects security issues. To make plug & work a success, it has to be ensured that only authorized devices are allowed to integrate themselves into the network. Appropriate security mechanisms have to be applied.

Self-organization based on peer-to-peer overlay networks

An important advantage of peer-to-peer networks is due to the fact that many requirements and properties of self-organization can be addressed that have been mentioned above. Peer-to-peer overlay networks strongly support the decoupling of hard and software. The strong link of an engineered application and hardware is today the main reason for inflexibility in automation systems. A peer-to-peer based engineering approach would overcome this disadvantage. For example communication links between PLC are today already configured in a peer-to-peer manner so that the automation domain can draw huge benefits out of the peer-to-peer technology.

Achieving self-organization properties is one objective of peer-to-peer overlay networks. These networks can store and automatically retrieve configuration data in and from an overlay network by using distributed hash tables (DHT). As an example will serve the replacement of devices in a plug & work manner. This process starts with the recognition of the device capabilities by checking some unique properties of the device. After that, the new configuration or the new firmware is downloaded by querying the peer-to-peer overlay network followed up by device reboot.

Positive feedback can be modelled by creating peer-to-peer networks. Those networks have no scalability limit because of centralized control and also no single point of failure [67]. This makes peer-to-peer technology a powerful tool having a good self-management in automation systems. Moreover, most the characteristics of self-organizing systems can be mapped to peer-to-peer systems. For instance, peer-to-peer networks show properties that no single peer has. This refers to emergence. In sensor and actor networks, peer-to-peer systems favor ad-hoc communication. Even small-world clustering according to similarity [68] can be modelled via peer-to-peer algorithms. Another possibility of profiting from peer-to-peer technology is semantic routing that uses information from previous search results. This enables the detection of information in the environment of the peer by listening to network traffic. [50], [51], [59].

This proves why peer-to-peer networks suit well for realizing self-organizing automation systems. In contrast to that, centralized control, like the application of servers, can not enable self-organizing structures by definition. As a conclusion, the peer-to-peer technology can foster the emerging self-organizing needs of future automation systems that come with low engineering efforts.
9 Conclusions and Next Steps

The early developments of the Internet protocol (IP) and its addressing scheme have been for a long time a synonym of the success of the Internet. With the growing number of nodes connecting to the Internet, it became clear that IPv4 addresses would soon run out. This problem is mostly due to the flat address scheme and assignment of IP addresses equally to both service providing (or servers) as well as routing nodes, and end-users. The use of IP addresses as a unique locator and identifier of the different nodes has shown its limitations of a growing Internet. Although IPv6 addresses were proposed at some stage, which seemed as the only way to solve the address problem, this has proved more difficult to deploy in practice.

The use of network address translators at the edge of the network have at least decreased the pressure coming from the fast growth of end-user nodes connecting to the Internet. This also meant that the problem of addressing can be split between private network addresses and globally accessible ones. This solution, however, no longer satisfies the addressing needs of the Internet of Things, where objects, end-devices, and machines are not just acting as clients but also provide services that are accessed across sub-networks and even globally.

In the manufacturing and industrial networks, where packet-switching and networking technologies have been adapted to this new application field, similar developments can be noted. Systems built on top of sensors, actuators, and controls have used IP addresses to identify end-nodes that connect to each other using TCP or UDP. The only difference to Internet applications was that industrial systems rely on well-planned resources that need to pre-configure all application interactions in advance. While adopting IP addresses in the factory floor, configuration tools used the new addresses as identifier of end-devices as this is the case in the Internet.

The problem with this approach is that identity of industrial devices differs from that of Internet servers. The Internet server normally hosts a bunch of services named symbolically in a human readable manner. This name is structured in domains that are broken down to identify services (e.g. web page is commonly known through its symbolic DNS name). The mapping of the names to the IP address is then done through DNS.

In industrial systems, device IP addresses proved to be insufficient as identifiers. Industrial devices are currently identified through a combination of a unique MAC address, an assigned IP address, and an assigned device symbolic name. This tuple also requires new tools to map the correctly planned IP addresses to MAC addresses and symbolic names. Also, application interaction planning requires heavy tooling to configure the whole system. Often, for this reasons, the auto-configuration mechanisms found in the Internet are simply turned off.

In order to enable a plug and play internet of things, it is important to understand the generic requirements for a naming and addressing system and the development within the Internet of Things. Also, it is essential to grasp the special needs of industrial and factory automation systems. Therefore, this document looks not only at the state of the art of IoT addressing and naming developments, but also at topics like configuration, discovery and directory systems. These problems are to some extent already solved, but the combination of these solutions needs to be more systematically understood.

The IoT@Work approach does not prescribe one or the other identifier scheme. Instead, the project targets generalized solutions, which might include any suitable technology. The generic solutions will be defined to fulfill the IoT Plug&Work
functions and their related requirements. In terms of relying on state of the art, in IoT@Work, we assume that systems engineers and process planners will still rely on a large range of criteria in selecting the appropriate technology or addressing scheme. We therefore have to define the methods to declare or define those choices. For instance the choice to use IPv6 for device addressing will require field-level devices and systems to support the technology. Also the system owner can only choose IPv6 depending on his/her owned address space. For instance, it is more difficult for a new manufacturing company to obtain large portions of public IPv4 addresses than an already existing company. This might force the new player to decide to adopt IPv6. Even the choice of protocols for addressing might be fundamentally different. For instance, using DHCPv6 instead of automatic IPv6 addressing is going to have major consequence on the industrial setup. Therefore, it is important to be able to compose with the technology decisions whether technical or socio-economical, in a more systematic manner.

IoT@Work targets those methods that enable intelligence within the network to both interpret the rules provided by the user, while mapping them to real devices and things.

Within the Plug&Work scenarios, we require a better understanding for bootstrapping. For this, we set the target of bootstrapping of reaching the productive state of each single device after start-up while taking into account both the needs for system programmers to plan their infrastructure in an offline process, and select some technologies to address or identify the system components, i.e., devices, networks, applications and services interacting through identities or symbolic names. This often costly process needs to be mapped in automatic steps to the connected physical network without manual configuration of each device. This step is called commissioning.

Commissioning according to IoT@Work uses network functions to discover the physical system and its details, on the one hand, and then relies on mapping functions that compare the discovered system details with those a programmer has defined.

---

**Figure 9.1: IoT@Work Naming and Addressing Function Split Across the Different Layers :** (the colour blue indicate the sub-layers where IoT@Work extensions are needed)
The interactions between different configuration functions and their dependencies are also another important aspect of a generic solution. The IoT@Work dependencies can be concentrated on the timely sequence of functions independently of any protocol choice or identifier scheme. This sequence is better understood in D2.2, but can be summarized as follows:

1. Detection of device connectivity and limiting access of a given device to resources.
2. Authentication of devices before allowing access to the network or other resources.
3. Use of network access control and some directory service to control the device attachment in the system. The directory service could be used to define both the methods and protocols used to address, and to identify the device. These might entail:
   a. Using the directory to guide the device through its configuration steps.
   b. Using the directory to guide the naming and addressing schemes chosen by the system administrators
   c. Using directory system to collect device descriptions, network locators, and other well structured semantic information to compare with planned and engineered semantics
   d. Mapping device IDs defined in the engineering process to the discovered device using semantics matching.
4. We expect at the service or application layer different types of names and different name spaces to coexist. Therefore, we will investigate a subset of how to support different names to distinguish difference services, similar to URIs, running with a single device.

In comparison to state of the art, the way we see IoT solutions to naming and addressing has different folds:

First, novel techniques are needed to define device semantics beyond electronic device descriptions (EDDs) or Field Device Tool (FDT) in a PROFINET or industrial devices in general. For instance, location, neighbouring or attached system (which machine is sensor A attached to), are important dynamic aspects, that pre-configured semantics do not cover today.

Also, components and granularity of systems is hard to capture by assigning IP addresses to each sensor and actuator. In IoT@Work devices are both field-level and networking devices connected to each other. It is in some cases important to distinguish each single sensor or actuator on their own through a unique ID (or address), even if it is only a component of higher-level complex machine. This depends on the assumed capabilities of these elemental components and the needs for “remote” or “third-party” applications to address these components individually. It is also worth understanding which components have to be distinguished through an addressable service interface and which are seen as part of the whole device and therefore will not be addressed on their own. Service-orientation in embedded systems might offer the right solutions for both cases.

Also the generalized approach using enterprise networking practices are targeted in particular, because they represent well established and understood standards that are not proprietary or specially developed for the sake of one engineering tool or the other.
In terms of naming and addressing, we will be exploring the potentials of IPv6 extensions to fit in the automation systems. Also, we would like to using IPv6 proposals for discovering and structuring locators of devices along their location, point of attachment and other dynamic semantics, which applies to IoT in general.

The upcoming steps are timed to design a bootstrapping architecture in the next phase, where both configuration requirements and name and addressing schemes are defined.

<table>
<thead>
<tr>
<th>Sub-task</th>
<th>Title</th>
<th>Objectives</th>
<th>Deliverable</th>
</tr>
</thead>
</table>
| T2.2     | State of the art on IoT addressing and naming applied to the factory | • State of the art of IoT addressing and naming  
• Naming and addressing needs seen from the factory field (besides the state of the art on tooling and existing practices) | D2.1 (M6) |
| T2.3     | Solutions for auto-configuring names and addresses within a bootstrapping architecture | • Auto-configuration prototyping for addressing and naming  
• Requirement assessment for a bootstrapping architecture  
• Architecture | D2.2 (M11) |
| WP1/T1.1 | Architecture requirements & assumptions | • Definition of concepts for Plug&Work  
• Implications on technical requirements WP2/3 | D1.2 (M14) |
| WP3/T3.4 | Solution for Security Bootstrapping | • Secure Plug&Work framework definition and mechanisms, considering usage of naming and addressing from a security perspective | D3.2 (M22) |

Table 9.1 – Follow-up Activities Based on This Deliverable
10 References


[2]. Völksen, Gerd; Müller, Jörg, Agent and peer-to-peer technologies in the context of cross-enterprise business processes, in the scope of ATHENA, European Research Project, 2004

[3]. Mahlmann, Peter; Schindelhauer, Christian; Peer-to-Peer Netzwerke; Springer-Verlag, Berlin, Heidelberg, 2007

[4]. Cox, Russ; Serving DNS using a Peer-to-Peer lookup service; MIT Laboratory for Computer Science, 2002

[5]. N. N.; Link Layer Discovery Protocol; Extreme Networks, USA, 2006

[6]. Sukhar, Ilya; Ramasubramanian, Venugopalan; Sirer, Emin Gün; A Peer-to-Peer DNS; Cornell University, 2006

[7]. Barthel, Alexander; Analysis, Implementation and Enhancement of Vendor dependent and independent Layer-2 Network Topology Discovery; School of Computer Science, Professorship of Computer Networks and Distributed Systems, Chemnitz University of Technology, 2005


[13]. Semantic Product Memory (SemProM) - URL: http://www.semprom.org/semprom_engl/ - Last checked: 2010-10-06

[14]. X.500 ITU Standards - The X.500 community site that is both a guide to the X.500 Standard and a repository for existing and new work being carried out on the standard.: http://www.itu.int/rec/T-REC-X.500/e


[19]. the official community gathering place and information resource for the UDDI OASIS Standard - http://uddi.xml.org/ - last visited 2011-11-09


[22]. VölkSEN, Gerd; MÜllER, Jörg. *Agent and peer-to-peer technologies in the context of cross-enterprise business processes*. In the scope of ATHENA, European Research Project, 2004


[27]. BRIDGE project. URL: [http://www.bridge-project.eu/](http://www.bridge-project.eu/) - Last checked: 2010-10-13


[31]. Electronic Device Description Language. URL: [http://www.eddl.org/Pages/default.aspx](http://www.eddl.org/Pages/default.aspx) - Last checked: 2010-11-09


[50]. Steinmetz, Ralf; Wehrle, Klaus. Peer-to-peer systems and applications. Berlin: Springer (State-of-the-art survey, 3485), 2005


[56]. Pussep, Konstantin; Oechsner, Simon; Abboud, Osama; Kantor, Miroslaw; Stiller, Burkhard Impact of Self-Organization in P2P Overlays on Underlay Utilization Multimedia Communications Lab, Technische Universität Darmstadt; Institute of Computer Science, University of Würzburg; Akademia Górniczo-Hutnicza im. Stanisława Staszica W Krakowie. In: Fourth International Conference on Internet and Web Applications and Services, 2009.

[58]. Liu, Lu; Xu, Jie; Russell, Duncan; Antonopoulos, Nick. **Self-organization of Autonomous Peers with Human Strategies.** School of Computing, University of Leeds (UK); Department of Computing, University of Surrey (UK). The Third International Conference on Internet and Web Applications and Services, 2008


[68]. Kobayashi, Hiroaki; Takizawa, Hiroyuki; Okawa, Takuro; Inaba, Tsutomu. **An Efficient Control Mechanism for Self-Organizing Overlay Networks of Large-Scale P2P Systems.** Information Synergy Center, Tohoku University, Japan; Graduate School of Information Sciences, Tohoku University, Japan; R&D Center, NTT East Corporation, Tokyo, Japan. In: Interdisciplinary Information Sciences, Vol. 13, No. 2, pp. 227–237 (2007)


[70]. CASAGRAS, an EU Framework 7 Project. **RFID and the Inclusive Model for the Internet of Things.** Casagras, July 2009.

[71]. IEEE 802 LAN/MAN Standards Committee. URL http://www.ieee802.org
[72]. ZigBee Alliance. URL: http://www.zigbee.org


[78]. DNS Service Discovery, URL: http://www.dns-sd.org/


[87]. EPCglobal, Inc. EPC™ Tag Data Standards Version 1.5. August 2010.

