# Taking a Legacy Interlocking to the Era of IoT

Bob Janssen Phd, AMIRSE, Siemens NL

## *SUMMARY*

*Early electronic interlocking systems were designed according to 1980's state of the art in information technology. Accessing diagnostic data to look under the bonnet of the interlocking would be cumbersome. Today's "Internet of Things" premise is to expose state data of sensor-equipped objects in the field. This paper explains how one can make a 1990's interlocking look like an IoT system. Authorised users can select and observe live state data from objects controlled by the interlocking, in IoT style. What's more, one can create a "digital mirror" of the interlocking system by providing rich data that represent both static configuration data and dynamic state data of the running system.*

## 1 INTRODUCTION

The first generation of Simis C family electronic interlocking (EIL) was built to control big yards such as Rotterdam Central Station. This EIL has been running with very little downtime since 1994 and by this virtue must be one of the world's longest running computer systems. The architecture of electronic signalling systems has evolved since; Interfaces are being standardised, control systems are being concentrated into national centres and ETCS caught on.

One newish trend in IT is colloquially called the Internet of Things which can be regarded as the kitting of field objects with sensors to gain a networked runtime view of the system as a whole. Strictly speaking, an EIL-controlled signalling system by this definition qualifies as an IoT system. This said, tools and technologies used in IoT outside railways have of course substantially evolved.

Older IT designs tend to store data in data silos that were accessible only offline or through proprietary interfaces. Diagnostic systems provided a tunnel view on a selected subset of live status data.

This paper shows how modern IoT tooling can be grafted on to such a system designed in the early 1990's.

Railway signalling systems are industrial control and command systems; controlling railway operations conceptually doesn't differ that much from controlling a large factory with its distributed sensors, actuators and process logics. As such, the signalling industry can benefit from developments in the realm of Command & Control systems. One such development is the OPC UA set of standards for modelling and accessing the data that describe a complex system (OPC Foundation, n.d.).

## 2 DATA CENTRED VIEW OF SIGNALLING

### 2.1 Relating data to systems

A system can be described in terms of data, e.g. a train has a weight, speed, position. One can break down the system into subsystems and to any desired level of detail, e.g. down to the temperature of the brake pads and to the date of latest maintenance. Some data are static properties, e.g. the make and date of manufacture whilst others are dynamic such as odometry. Data have a meaning that needs a precise description. For instance, temperature: where is it measured, with what sensor, which units, when was the measure taken, what precision and in what kind of variable is it stored, integer or float. OPC UA provides a standardised model for defining a so-called address space with constructs for defining data types, arrays, variables, objects and relations. This address space can be designed to fully inform an end user about the system's data set.

System data that have been cast into an OPC UA address space can be exposed by a server. In a typical industrial setup, a PLC that controls a process, e.g. a boiler on factory floor, has an embedded OPC UA server that exposes state data. A client can, *without prior knowledge of the address space, discover and consult the data*. This approach is proven to be very useful for building off-the-shelf SCADA systems. Such supervisory systems can pick and mix a subset of the diagnostic data into a view that is of interest to particular use cases.

From the point of view of an interlocking, a railway yard is a connected set of field elements such as points, signals, track vacancy sections and level crossings. A signal is an object that is composed of a set of lamps and a lamp has a colour and can be on or off. The topology describes the connections between field elements. This wording is of course chosen to bridge the gap between the classic view of an interlocking as a control system and the data-centric view.

Simis electronic interlockings treat field elements as software objects with a live state vector that is stored in memory. A state vector is a list of variables such as the occupancy state of a section, the position of a point and the on/off status of a signal lamp. The state vector is stored in memory and is part of the control process. The objects also have static properties such as identity, geographical position of a signal, length of a section, name and number and relations to other objects. The interlocking is of course aware of topological relations that allow it to build routes through the network. Therefore, the EIL knows exactly which sections must be vacant and which points must be in what positions before a signal may open.

Dynamic state vectors, static properties and topological relations can all be exposed through an OPC UA address space.

## 2.2 Opening the Simis C data silo

The Simis C EIL is composed of a number of computer nodes that exchange state data via a data bus. Telegrams on this bus are logged in a so called *Busmithörrechner* (BMR). We tapped into this BMR to extract data as they're logged in semi-real time. The telegrams are parsed, de-marshalled and the read values are stored in OPC UA objects that mirror the objects known in the interlocking. By doing this, we've created a *digital mirror* of the running system.

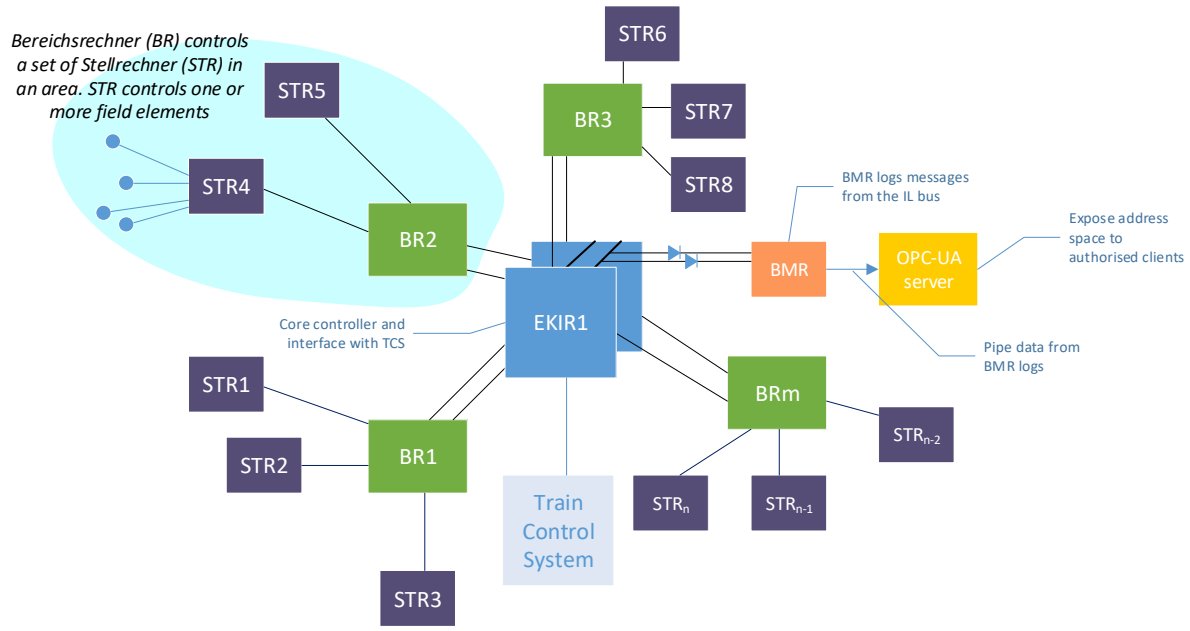


*Figure 1: conceptual graph of the communication nodes in and around the interlocking*

The BMR logger and the server are separate machines because the original BMR logger could not be modified but fortunately, it provides access through an SSH interface.

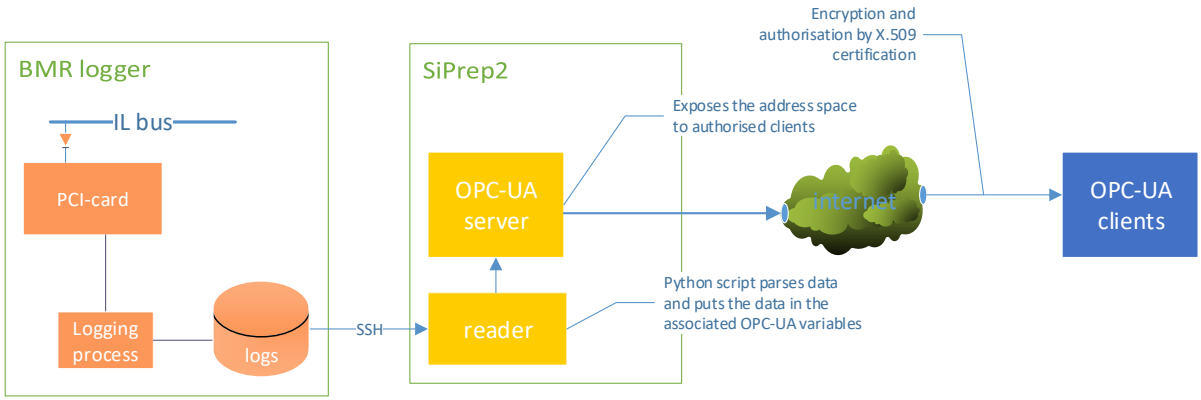


*Figure 2: The BMR logs interlocking data that the OPC UA server exposes*

The server presents the signalling objects and their attributes. When the *reader* in Figure 2 reads state changes from the bus telegram, it updates the associated object in the OPC UA server. A client can subscribe to a selection of attributes, e.g. failed signal lamps.

## 2.3 Defining security and roles

A data diode physically separates the BMR from the vital computing nodes of the EIL. In this case, this was as simple as cutting the sending fibre from BMR towards the IL-bus. Thus, even if the BMR were compromised, the IL remains inaccessible. Yet, we consider that the data from the IL-bus should be access-controlled to a very high standard. Fortunately, the OPC UA standards and toolboxes provide encryption and authorisation out-of-the-box by means of X.509 certificates, well known in Public-Key Infrastructures (PKI). Only clients that present known public keys to the server can read data. Only when the client owns the private key that is paired with the public key, can the client decode the messages.

OPC UA provides mechanisms to tailor the views, i.e. the subsets of data that a client can see to given use cases. This allows us to create roles such as maintainer or analyst where the maintainer gets information about active failures plus information required to fix the failure whereas an analyst could get to view a wider range of data that allow deep analysis over longer time frames.

The choice to separate the existing BMR logger from the new OPC UA server is also motivated by our desire to have low coupling between the existing and new equipment. Proving that one subsystem has no adverse effect on the other is substantially easier when the interface is “thin”. This also allows future developments like migrating the OPC UA service to a cloud server. Judiciously inserting abstraction layers between subsystems is a corner stone of the Reference CCS architecture (EUG and EULYNX partners, 2019).

The new machine is accessible via a Secured Shell (SSH), using X.509 certificates. In order to ward of future attacks that may exploit first day vulnerabilities, this machine is kept up to date with safety patches, a procedure that would be very hard to do if the machine were part of a legacy safety case that prescribes a rigid and time consuming process before modifications can be applied. Timely applying security patches is obviously essential which motivated our choice to exclude the new machine from the safety case.

## 2.4 Aggregating and using data

What can a client do with the data that the EIL exposes? A legacy diagnostic system offers a keyhole view on a subset of live data that were selected during the system design, back in the late 1980’s. These legacy systems were originally line printers, and later PC screens displaying a selection of error messages. An OPC UA client can theoretically view the complete live status vector, if the status data can be gleaned from the EIL data bus.

A typical use case is a remote client that wishes to monitor the delay between point throwing delays and section occupancy delays. This allows a maintainer to receive an alarm when delays start to creep, indicating imminent hardware failure.

Below figure shows an off-the-shelf OPC UA client browsing information from a running server. Note that the objects are dynamically updated when the reader in Figure 2 parses a telegram reporting status change of a variable that can be allocated to an object's attribute.

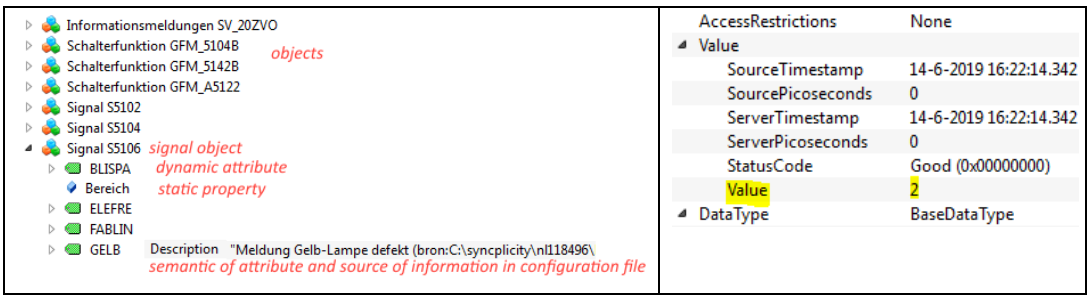


*Figure 3: screenshot of an OPC UA client browsing data from information from an OPC UA server. The client browses and reads actual data values and the semantics: no prior knowledge is needed of the data that the system serves.*

## 2.5 Enriching data

The server's address space can enrich the object's dynamic status vector with static configuration data. The attributes "GELB - yellow lamp of Signal S5102 failed" shown in Figure 3 may still seem cryptic and terse. This is where one can enrich the address space with information to support maintenance. Presently, staff need to locate the signal and consult paper documents to trace the relay house, cabinet and rack where the failure would be located. This latter information is static but is easily added to the OPC UA address space. The address space can mix dynamic information with static configuration data as needed.

For this purpose, one can analyse the available configuration data and one wishes to expose on the server. Below UML class model represents the data that is associated with object failures.

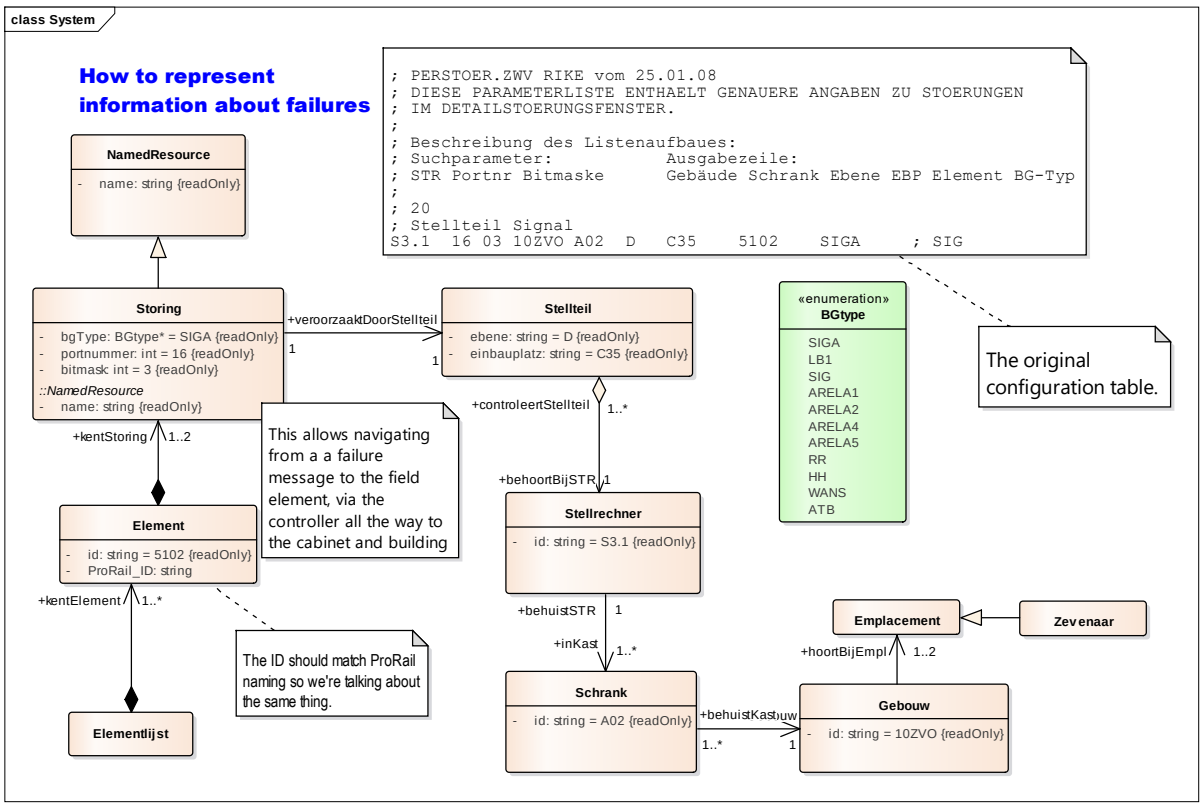


*Figure 4: A UML class model representing static information used to enrich failure messages*

An OPC UA server can expose static data that is structured according to this model *alongside* the dynamic object information shown in Figure 2. An authorised OPC UA client can, without prior knowledge of the system, consult the value and the semantics of the object and browse for further relevant information.

Mixing knowledge of the system with its dynamic state data is very powerful because it informs remote users like maintainers about the context of the data without having to resort to paper plans and documents that are typically offline and may well be out of date.

### 2.5.1 Presenting data tailored to the end user

End users can run an OPC UA client on their laptop that consults the server's address space. The client can wrap the information in a purpose-built user interface that offers only the required subset of information.

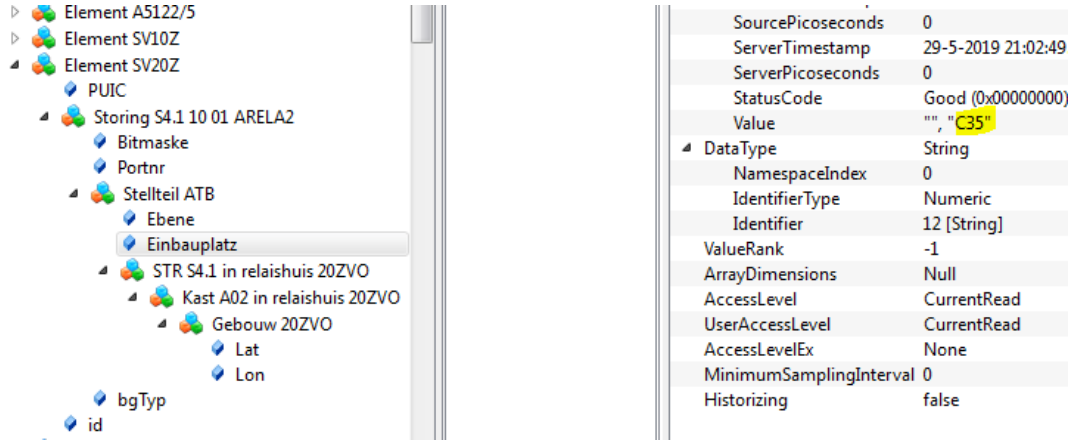


*Figure 5: screenshot of an off-the-shelf OPC UA client from Unified Automation browsing static data reflecting the structure, value, and semantics of a data point.*

At present, the interlocking's diagnostic system merely informs about a failed signal and the responsible service person, piecemeal informed by the signaller, must consult paper plans on site before he can act. Keeping paper plans up to date in a big yard with distributed substations is notoriously hard.

In an automated process, the responsible service person retrieves the information directly from the OPC UA server. For instance, an app running an OPC UA client would notify an error message from a failed signal combined with static configuration information about the equipment controlling the signal. A nicely designed wrapper around the OPC UA client can present exactly the information that service personnel needs to quickly intervene. Previously, the designer of a client to a diagnostic server needed a detailed interface specification but now, an app that runs an off-the-shelf OPC UA client, like the one shown in Figure 5, needs no prior knowledge of interfaces because the OPC UA server provides the full design of the communication interface plus the design of the data values, semantics and variable types out of the box. In other words, any third-party software designer can focus on building an app around the OPC UA data.

## 2.6 Further use cases

Another exciting future use case concerns simulation programs. We can automatically convert the available configuration files like topology of the rail network, signal identities, configuration of relay rooms, etc. into the server's address space. Henceforth, a user needn't painstakingly acquire configuration from paper plans and lists but can simply consult the data exposed by the server. Thus, one can construct a simulation model that faithfully mirrors the real yard because it was built with the data that are extracted from the original configuration files and exposed via the system's OPC UA server.

The mix of static configuration data and dynamic status data allows clever algorithms to quickly answer queries like

- What routes are affected when point x fails?
- What routes are still available when the yellow lamp on signal y fails?

- When point Z is clamped, which routes will lack flank protection?
- When a cable is cut, what parts of my rail network will be knocked out?

Finally, one can look over the horizon; once the system has been proven in action for some time, one can consider creating an OPC UA view of the network tailored to drivers or signallers. This could offer a backup signalling system, should the light signalling or transmission of information to onboard ETCS displays fail. After all, the OPC UA server directly acquires live data from the interlocking, including a view of point positions and section occupancies and this can prove a useful support for, non-SIL, on sight navigation through the rail network in case of a break in the link from interlocking to driver's ETCS display.

# 3 CONCLUSION

Electronic interlockings keep dynamic information about field elements in a silo. Static configuration information is kept in another silo. OPC UA servers can safely expose all this information to authorised clients. The mix of this system knowledge and state data allows easy access for a wide range of use cases that up to now involve painstaking collection of data from files and documents that are so far locked in.

Siemens Mobility NL is currently testing this system for the Simis C interlockings.

On a final note, one may question the fact that this approach has been tailored to a legacy interlocking. But in fact, EULYNX control modules will come with OPC UA out of the box. This suggests that the approach is here to stay.

# 4 REFERENCES

EUG and EULYNX partners, 2019. *RCA Alpha – Architecture Overview,* Bruxelles: EUG and EULYNX.

OPC Foundation, n.d. *OPC technologies - Unified Architecture.* [Online]
Available at: https://opcfoundation.org/about/opc-technologies/opc-ua/
[Accessed 14 6 2019].