ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA162

# MONITORING OF CERN'S DATA INTERCHANGE PROTOCOL (DIP) SYSTEM

B. Copy, E. Mandilara, I. Prieto Barreiro, F. Varela Rodriguez
CERN, Geneva, Switzerland

*Abstract*

CERN's Data Interchange Protocol (DIP) [1] is a publish-subscribe middleware infrastructure developed at CERN to allow lightweight communications between distinct industrial control systems (such as detector control systems or gas control systems).

DIP is a rudimentary data exchange protocol with a very flat and short learning curve and a stable specification. It also lacks support for access control, smoothing or data archiving.

This paper presents a mechanism which has been implemented to keep track of every single publisher or subscriber node active in the DIP infrastructure, along with the DIP name servers supporting it. Since DIP supports more than 55,000 publications, regrouping hundreds of industrial control processes, keeping track of the system activity requires advanced visualization mechanisms (e.g. connectivity maps, live historical charts) and a scalable web-based interface to render this information is essential.

## DATA INTERCHANGE PROTOCOL (DIP)

DIP [1] is a communication system which allows relatively small amounts of soft real-time data to be exchanged between very loosely coupled heterogeneous systems. These systems do not need very low latency. The data is assumed to be mostly summarised data rather than low-level parameters from the individual systems, i.e. cooling plant status rather than the opening level of a particular valve.

DIP publications contain :

- a key-value map supporting standard basic data types (such as string, integer *etc.*.) or their array-based variants (string array, integer array *etc.*.),
- a publication timestamp indicating when the publication data was issued,
- a quality flag indicating over two logical bits the confidence the data publisher places in the issued publication update
- an optional quality string, giving further details about the reason for a lack of confidence in the issued publication update (*e.g.* sensor out of range).

DIP is a peer-based data exchange protocol : peers (publisher and subscriber) locate each other via a naming directory (hereby referred to as a DIP Name Server, or DIPNS), then establish a direct TCP communication based on the DIM protocol [2].

DIP publisher and subscriber processes locate each other on the network via a so-called DIP Name Server (DIPNS), which acts as a directory and prevents publication naming collisions. Among many usages, DIP is employed for essential, non-critical communication such as the Large Hadron Collider (LHC) Handshake sequence that allows the LHC machine and LHC experiments (via the JCOP framework) to initiate data acquisition sequences.

This paper presents the last two web-based DIP services recently introduced at CERN: the DIP Contract Monitoring System and the DIP Web Tools.

## DIP CONTRACT MONITORING SYSTEM

DIP is an open and permissive data exchange protocol: it does not provide any access control on data, allows data to be pushed at any supported rate without support for smoothing or filtering, and does not provide any history of its participants' activity. Such a permissive approach to data exchanges requires however a good understanding of the current state of the entire system: DIP data providers must be able to know which other computers and processes are currently consuming their data; DIP data consumers must be able to understand simply why the data they are relying on might be missing from the infrastructure, and for how long it has gone missing; all DIP users must be able to see the level of availability of the DIP name servers.

The DIP Contract Monitoring (DIPCM) is an application that fulfils two main objectives:

- to provide a widely accessible interface to DIP publications,
- and to help specify and enforce quality constraints on DIP publications for monitoring purposes.

Quality constraints placed on DIP publications are gathered into a **DIP contract.**

In order to implement these objectives, it was decided to adopt the CERN Monitoring Data Entry System for Technical Infrastructure (MoDESTI) [3] as the system to gather the quality constraint specifications from end-users, and run said specifications through an approval and signature workflow, thereby ensuring that all involved parties are in agreement over the contract.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA162



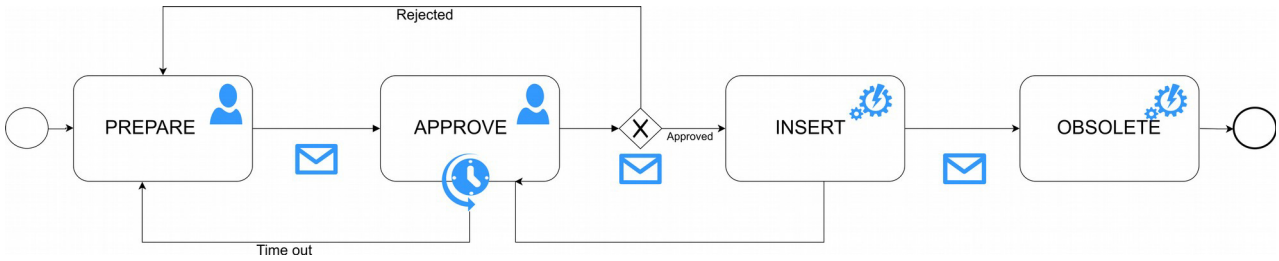


Figure 1 : DIP Contract registration workflow.

Another important part of the system is the CERN Control and Monitoring Platform (C2MON) [4] engine. C2MON specialises in acquiring data from industrial control systems present at CERN, including DIP, and, in addition to repropagating this data in a robust and scalable manner, also maintains availability and data quality statistics, which are at the heart of our DIP contract monitor.

## *User Interface & Functionality*

The DIP Contract Monitoring system presents itself to end users as a web application. It offers two basic functionalities:

- The registration of a new publication contract.
- The consultation and modification of all active publication contracts.

By definition, a **contract** is a list of publications a service provider pledge to publish. An example is LHC experiments subscribing to publications provided by the CERN accelerator infrastructure can be captured in a **contract**. As illustrated in **figure 1**, in order to register a new **contract**, a CERN user has to go through a three step workflow (the OBSOLETE phase being only used by administrators to retire the contract), where they are asked to submit the essential information and metadata of their contract, *i.e.* :

- name and description,
- starting and expiry date of the contract,
- the list of desired publications to subscribe to as well as their complete data type specification.

Upon successful completion of the wizard, when all fields and user selections are valid, the **contract** to be registered is pushed to MoDESTI. From this point on, MoDESTI's workflow engine (based on the open—source business process management engine Activiti [5]) takes care of the authorization and propagation of the contract registration.

To view all contracts which a given user or groups of users has created and update one, a second user interface is provided, which displays all contracts in a table, containing meta-data such as the name of the creator, the date of its creation, its publishers and subscribers and its status. Any modification to the contract will invoke the DIP Contract Monitoring System workflow, notifying contract stakeholders that their approval is once again requested.

## *Backend Architecture*

DIP Contract Monitoring System is a Spring Boot [6] application. It is based on the concept of a simple workflow that backs the application's front-end user interface through the same aforementioned three basic stages: **registration**, **approval**, and **insertion into a C2MON configuration database**.

The first stage is the preparation of a contract, which is done as evoked earlier by visiting the DIP Contract Monitoring System web application and filling all the required fields of the wizard. At the end of this procedure, the second stage, **approval**, is triggered.

At the beginning of this **approval** step, a notification is dispatched to the stakeholders that are responsible for the individual publications included to a contract. These stakeholders are invited to review the contract details in MoDESTI and approve or reject the requested contract. If the contract is rejected, its creator needs to edit the information they provided as per the approver's comments. If the request is approved, it progresses in the workflow to the final stage, which is the configuration of the contract into C2MON [4], which will allow it to be actively monitored.

## *Implementation in MoDESTI and C2MON*

To address different use cases, MoDESTI supports modular plug-ins. Each MoDESTI plug-in contains a schema describing the data domain and its constraints, and also provides a Business Process Model and Notation (BPMN) workflow specification.

In the case of DIPCM, a dedicated plug-in was created in order to cover the application's workflow as described earlier.

C2MON has a persistent storage mechanism implemented with Elasticsearch, that is used to store alarms or other data that are wanted to be monitored, under the interface of a **data tag**. The **data tag** is a C2MON Tag coming from an external data source, while a C2MON Tag is an object destined to be updated, logged, published to clients and used in data views. It

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA162

corresponds to a single external data point, represented by a Java primitive type (String, Float, Integer, etc.). Every **data tag** has to provide a so-called "hardware address", which contains the information needed for subscribing to this data point. After this subscription, and with the necessary Data Acquisition (DAQ) module in function, updates about data changes can be coming in and stored into Elasticsearch.

## DIP WEB TOOLS

Since DIP enables the real-time communication between heterogeneous systems, it is being widely used and the demand for more data exchange is continuously growing. At the same time, the state and flow of information is often ignored while they are not negligible. Information exchanges therefore need to be monitored so that, for instance, invalid publications can be detected, invalid name formats can be reported upon and in general, to keep track of what information is being published.

For this reason, four web-based tools were implemented:

- DIP Web Browser
- DIP Publishers Monitor
- Connectivity Maps
- DIPNS Monitor

### *DIP Web Browser*

The DIP Web Browser simplifies the access to DIP data from anywhere (inside the CERN network); through a high-performance rendering tree featuring all the existing DIP publications, users can select the publications they are interested in, and display the online data as well as their quality status and last update timestamp.

### *DIP Publishers Monitor*

The DIP Publishers Monitor is a tool that aggregates information about publishers in DIP (publisher name, process id, current state, host, total number of services & DIM version), and also visualizes all the connections of each publisher as well as its availability history.

### *Connectivity Maps*

The Connectivity Maps provide a visualization of all ongoing connections in a subset of DIP publishers and subscribers. They also provide detailed information about each of these publishers (publisher name, process id, current state, host, total number of services & DIM version), as well as the publisher's availability history.

### *DIPNS Monitor*

The DIP Central Name Server (DIPNS) Monitor provides visualizations about the CPU and memory usage, the number of publications, clients and servers of DIP Central Name Server.

### *Common Implementation Details*

The front ends of all these tools were created making use of Polymer Web Components [7], which allows to create reusable widgets or components, and very simply reuse them in multiple web pages. Components for the data exchange (for the LHC broadcast [8] and AJAX requests), as well as the connectivity maps, charts etc. were encapsulated throughout all the tools, while they interoperate via standard JavaScript DOM events. Powerful JavaScript libraries were also used for the implementation of the components: Data-Driven Documents (D3.js) [9], Highcharts, jQuery, Packery, and the Atmosphere framework [10].

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THPHA162



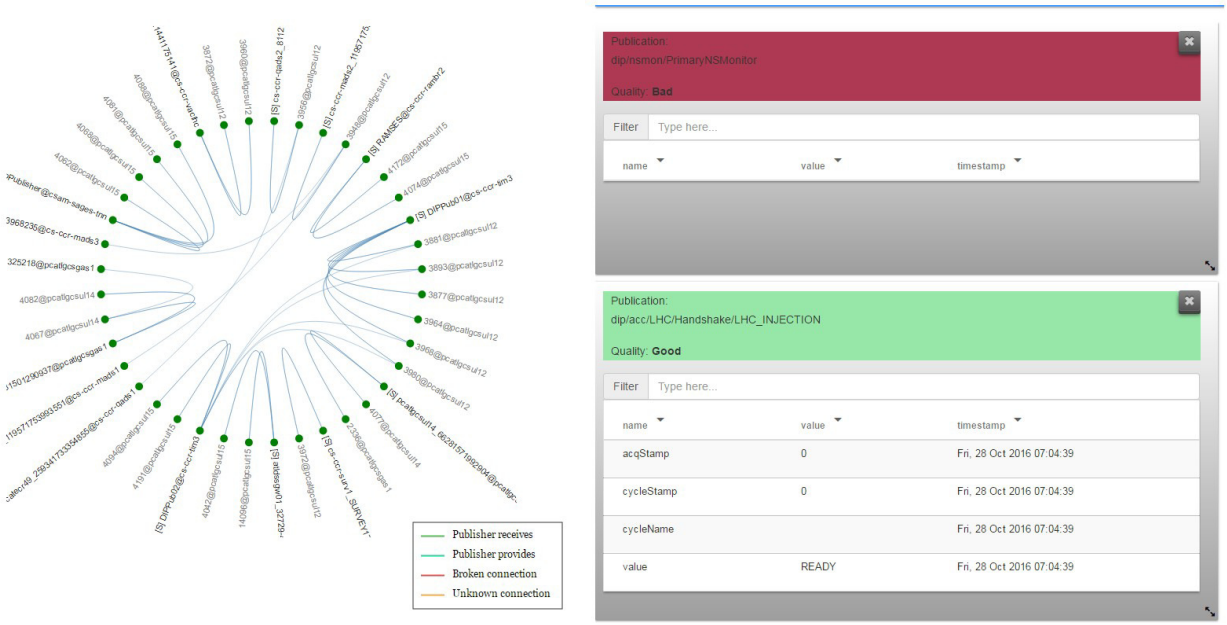


Figure 2 : Connectivity map and dynamic data table web componentexamples.

**Figure 2** presents examples of a D3 connectivity map visualization, as well as a dynamic data table.

WebSockets and WebRTC enable the distribution of industrial controls data (such as coming from devices or SCADA software) in a scalable and event-based manner. A web component that encapsulates the LHC broadcast mechanism [8] was implemented with objective to connect to a broadcasting server and by subscribing to the interesting publications to receive real-time updates and direct them to the frontend.

## REFERENCES

[1] W. Salter *et al.*, "DIP Description" LDIWG (2004), https://cern.ch/dip/.

[2] C. Gaspar *et al.*, "DIM, a portable, light weight package for information publishing, data transfer and inter-process communication", Computer Physics Communications 140 1+2 102-9, 2001.

[3] R. Martini *et al.*, "Tools and Procedures for High Quality Technical Infrastructure Monitoring reference Data at CERN", WEPGF141, Oct 2015 , ICALECPS'15, Melbourne, Australia.

[4] M. Braeger *et al.*, "High availability monitoring and big data : using Java clustering and caching technologies to meet complex monitoring scenarios", MOPPC140, Oct 2013, ICALEPCS'13, San Francisco, USA.

[5] T. Rademakers, "Introducing the activiti framework", in *Activiti in action*, Shelter Island, NY, USA, Manning publications, 2012.

[6] C. Walls, "Deploying Spring Boot applications", in *Spring Boot in action*, Shelter Island, NY, USA, Manning publications, 2015.

[7] D. Glazkov and H. Ito, "Introduction to Webcomponents", 24 July 2014, http://www.w3.org/TR/components-intro/

[8] B. Copy *et al.*, "Scalable web broadcasrting for historical industrial controls data", WEPGF042, Oct 2015, ICALECPS'15, Melbourne, Australia.

[9] S. Murray, "Introducing D3", in *Interactive Data Visualizations for the web*, Sebastopol, CA, O'Reilly Media, 2017.

[10] I. Hickson, "The WebSocket API", W3C Consortium, 20 September 2012, http://www.w3.org/TR/2012/CR-websockets-20120920.