Optimal Construction Management & Production Control

D2.7 – API Specification for the Digital Twin Platform

WP2 – Digital Building Twin Platform

Version 1.0

Issue date:	30/06/2022
Author(s):	Sébastien Bolle (Orange), Maria Teresa Calcagni (UNIVPM), Yasmin Fathy (UCAM), Nicolas Bus (CSTB), Jonas Schlenger (TUM), Timson Yeung (TECHNION), Dennis Pawlowski (RUB), Karsten Winther Johansen (AU)
Editor:	Orange
Lead Beneficiary:	TUM
Dissemination level:	Public
Type	Specification
Reviewers:	Jonas Schlenger, André Bormann (TUM), Nicolas Bus, Guillaume Picinbono (CSTB), Yasmin Fathy (UCAM), Timson Yeung (TECHNION), Maria Teresa Calcagni (UNIVPM), Dennis Pawlowski (RUB), Karsten Winther Johansen (AU)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 958398

EXPLANATIONS FOR FRONT PAGE

Author(s): Name(s) of the person(s) having generated the Foreground respectively having written the content of the report/document. In case the report is a summary of Foreground generated by other individuals, the latter have to be indicated by name and partner whose employees he/she is. List them alphabetically.

Editor: Only one. As formal editorial name only one main author as responsible quality manager in case of written reports: Name the person and the name of the partner whose employee the Editor is. For the avoidance of doubt, editing only does not qualify for generating Foreground; however, an individual may be an Author – if he has generated the Foreground – as well as an Editor – if he also edits the report on its own Foreground.

Lead Beneficiary of Deliverable: Only one. Identifies name of the partner that is responsible for the Deliverable according to the BIM2TWIN DOW. The lead beneficiary partner should be listed on the front page as Authors and Partner. If not, that would require an explanation.

Internal Reviewers: They should not belong to the authors. They should be any employees of the remaining partners of the consortium, not directly involved in that deliverable, but should be competent in reviewing the content of the deliverable. Typically, this review includes identifying typos, identifying syntax and other grammatical errors, altering content, adding or deleting content.

BIM2TWIN KEY FACTS

Project title	BIM2TWIN: Optimal Construction Management & Production Control
Starting date	01/11/2020
Duration in months	42
Call (part) identifier	H2020-NMBP-ST-IND-2020-singlestage
Topic	LC-EEB-08-2020 Digital Building Twins (RIA)
Fixed EC Keywords	-
Free keywords	Digital Twin; Graph database; BIM; IA; Machine Learning; Image recognition; process optimization; safety improvement
Consortium	17 organizations

BIM2TWIN CONSORTIUM PARTNERS

	Partner	Country
1	CSTB: CENTRE SCIENTIFIQUE ET TECHNIQUE DU BATIMENT	FR
2	TECHNION: ISRAEL INSTITUTE OF TECHNOLOGY	IL
3	UNIVERSITY OF CAMBRIDGE	UK
4	TUM: TECHNISCHE UNIVERSITAET MUENCHEN	DE
5	INRIA: INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE	FR
6	FIRA GROUP OY	FI
7	INTSITE LTD	IL
8	FUNDACION TECNALIA RESEARCH & INNOVATION	ES
9	ACCIONA CONSTRUCCTION SA	ES
10	RUHR-UNIVERSITAET BOCHUM	DE
11	SPADA CONSTRUCTION	FR
12	UNIVERSITA POLITECNICA DELLE MARCHE	IT
13	UNISMART – FONDAZIONE UNIVERSITÀ DEGLI STUDI DI PADOVA	IT
14	ORANGE SA	FR
15	SIEMENS AKTIENGESELLSCHAFT	DE
16	IDP INGENIERIA Y ARQUITECTURA IBERIA SL	ES
17	AARHUS UNIVERSITET	DK

DISCLAIMER

EXECUTIVE SUMMARY

BIM2TWIN aims at proposing an innovative approach for the building construction process by bringing a complete knowledge of the real situation of the work and by sharing between stakeholders a thorough situational awareness. The BIM2TWIN approach relies on a Digital Building Twin Platform (DBTP), fed by several and heterogenous data sources (building elements, construction activities and tasks, monitoring data, project scheduling, resource location and usage, etc.) and feeding the business applications used during the construction process. This data-driven approach is based on an enrichment process from raw data to information and knowledge for better decision making.

The DBTP aims at sharing the knowledge between construction project stakeholders. The Digital Twin must correspond directly to the building design and the defined construction schedule so that everyone involved can access reliable, accurate, real-time information on the project's status and essential information for coordinating their work with others.

To reach this goal, the DBTP API (Application Programming Interface) is an essential enabler to make the connection between the different WPs and business applications. On one side each application can use the API to benefit from the knowledge in the Digital Twin to enrich its business process and evaluate impact of an event occurred in another business process. On another side each application can produce more accurate data and information and use the API to push it in the Digital Twin and make it available to the other business processes.

The DBTP API provides to the business applications the mean to access and update the graph-based DT. It is structured with the BIM2TWIN ontology and organized in three data layers: the Raw Data Layer (the data lake), the Logical Data Layer (the mirror of physical objects, processes and persons) and the KPI Layer (the high-level indicators). The graph structure of the DT enables to interlink data and information from different business not only inside a layer but also between the different layers. This interlinking based on the graph relations are valuable and the foundation to make the connection between the business applications and processes. It is also helpful for estimating the impact for a business of an event in another business and its propagation.

The API specification covers the following business applications: Progress Monitoring and Quality Control, Safety Management, Equipment Optimization, Project Planification, KPIs Management and Project Management.

The dissemination of the DBTP API specification in a standardization organization, an open forum or an open-source community should be studied in the activities of WP9 (Dissemination, standardization, exploitation).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Abbreviations

API	Application Programming Interface
B2T	BIM2TWIN
BIM	Building Information Model
BLOB	Binary Large Object
DBT	Digital Building Twin
DBTP	Digital Building Twin Platform
DT	Digital Twin
GPS	Global Positioning System
IFC	Industry Foundation Classes
IoT	Internet of Things
KPI	Key Performance Indicator
LIDAR	Light Detection and Ranging
LOD	Level of Details
REST	Representational State Transfer
TOF	Time of Flight
WP	Work Package

1 INTRODUCTION

1.1 Scope and Objectives

The objective of the BIM2TWIN (B2T) project is the design and development of a Digital Building Twin Platform (hereinafter called DBTP) to improve efficiency in the management of building construction processes. BIM2TWIN proposes a comprehensive and holistic approach for creating a Digital Twin for building construction. In particular, it will provide real-time status of products and processes on-site and throughout the entire building life-cycle, including progress and quality of the work, location of workers, equipment and materials, safety conditions, etc. The Digital Building Twin (hereinafter called DBT) must be able to capture the physical state of the building and the state of the construction process as it is. It must correspond directly to the building design and the defined construction schedule so that everyone involved can access reliable, accurate, real-time information on the project's status and essential information for coordinating their work with others. This will ensure that managers are aware of the status of the product, process and overall project. This requires understanding key performance indicators and situational awareness issues during the construction processes, as well as identifying opportunities for improvement in terms of safety, work progress monitoring, optimal resources utilization (including equipment and people) and scheduling.

The objective of this document is to provide the API specification of the graph-based DBTP for B2T. The B2T platform is based on the research platform provided by Orange: “Thing in the Future” (Thing’in for short). Thing’in enables the management of digital twins of systems and complex systems (systems of systems) thanks to its dual modelling capabilities: property-graph modelling and semantic modelling. As described in [4], Thing’in has a dual modelling capabilities: property-graph modelling and semantic modelling - enabling the efficient management of digital twins of systems and complex systems (systems of systems). The API described in this document will rely on the generic API of Thing’in and are aimed to be exposed as business enablers on top of the generic graph-based platform.

1.2 Relation to other Work Packages

In B2T project, the DBTP is provided by WP2 and it has a central role as all the Application Work Packages (WP3 to 7) will rely on it to implement and provide their Digital Twin Services.

Inside WP2 itself the DBTP has also a central position as shown in Figure 1. The DBTP platform is developed and provided by the Task 2.5, which is fed by Task 2.1, providing the requirements for the DBTP. Task 2.2 mainly takes the project intent data (BIM, schedule, resource assignment, etc.) and convert this information so that it follows the data models that B2T defined in T2.1. Task 2.3 provides the integration of the construction monitoring tools into the Digital Twin. Task 2.4 proposes the tools to transfer information and knowledge from the Digital Twin of the construction phase to the platforms and tools that will be used for the operational phase of the building. Task 2.6 provides the B2T Dashboard and it will rely on the services exposed by the DBTP.

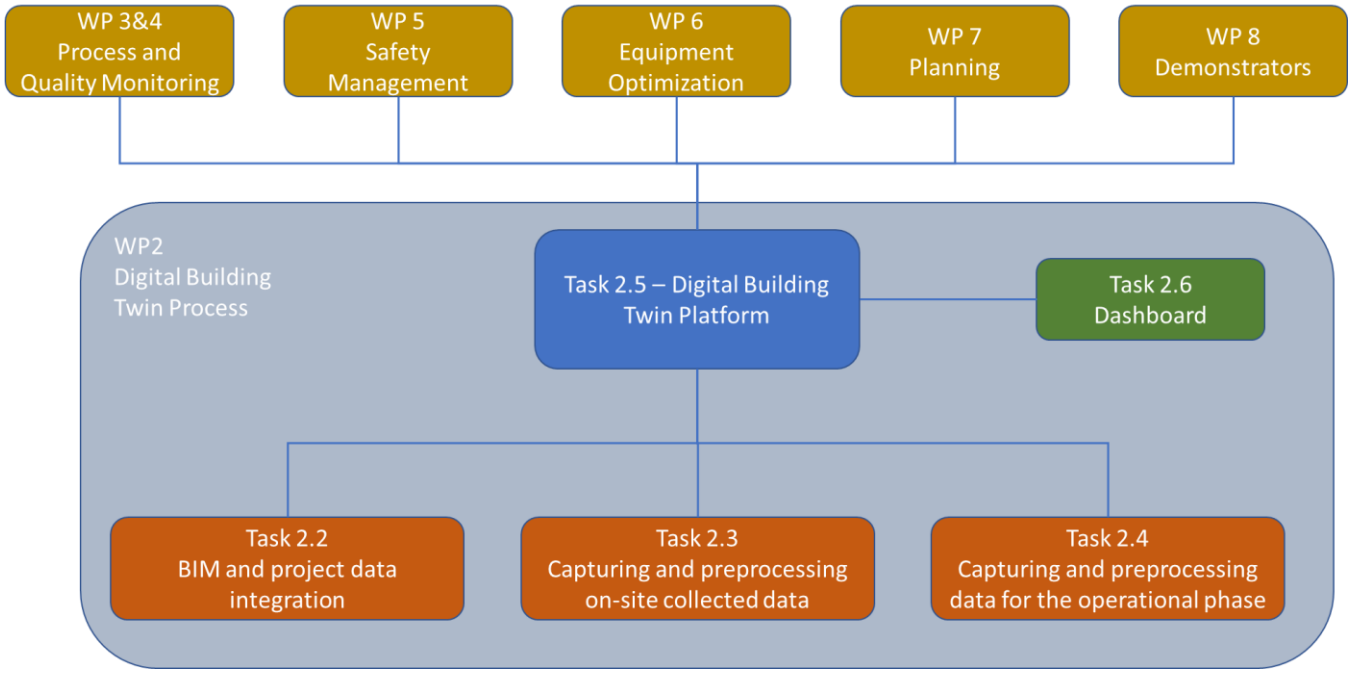

Figure 1: Global view of the project and relationships between Work Packages

1.3 Structure of the document

The document is organized as follows. Section 2 explains the benefit of the graph approach for the B2T digital twin and the benefit of sharing the knowledge on the building construction project. Section 3 describes the B2T graph organization in its different layers, exploited by the API. In Section 4, we describe the API for the requirements of each WP to achieve the overall objective of the B2T project.

2 ESSENTIAL REQUIREMENTS OF THE GRAPH-BASED DIGITAL BUILDING TWIN

The Building Digital Twin Platform developed by B2T is based on a graph representation where each system is connected and interacts with other systems based on a semantic description. Graphs are the most universal, versatile and adaptable way to describe complex systems and represent their (hierarchical) relationships and dependencies at multiple levels and scales (systems and systems of systems) [10][11][12][13]. On the other hand, semantics brings a formal description of the entities through ontologies. Ontology describes the semantics of the data, improving interoperability by providing a uniform way to enable (complex) systems to understand, communicate and exchange information with each other.

The graph structure of the digital twin can be enriched by embedding contextual information to the graph's elements (i.e., nodes). An information provided in one node of the graph is enriched by the relations which exist between this node and other nodes in the graph where nodes relations can be inferred through direct or indirect connection between nodes. In principle, the relations can also provide their semantic: "isIn" to describe this element is inside a room of the building, "actsOn" to describe this IoT actuator allows to perform an action on such building element.

Figure 2 provides an example of a graph representation of a building where two graph nodes are representing two neighboured rooms of the building and are linked with an "isAdjacent" relation. The first room includes a temperature sensor which is represented by a "hasTemperatureSensor" relation to the room. The second room is in relation with an equipment producing a lot of heat. Without the graph an application would receive a high temperature from the sensor without being able to explain why it may be abnormal. Thanks to the graph, the application will benefit from the capacity to analyze the high temperature value in its context and bring a more accurate explanation of the situation. Also if there is a failure in a temperature sensor the application will be able to analyze the cascading effects of such failure.

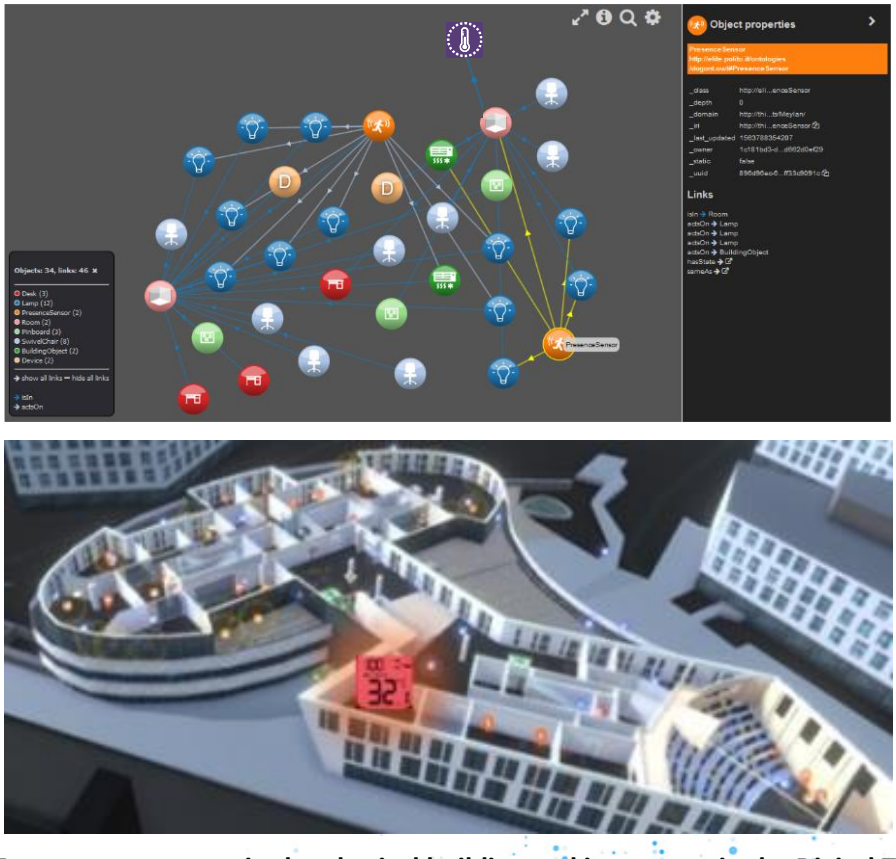

Figure 2: Temperature sensor in the physical building and its context in the Digital Twin graph

Similarly, suppose a wall has a quality issue (e.g., a discrepancy between the actual and intended designs). In that case, a contractor will be able to identify the adjacent objects/elements that might need to be checked by querying the graph structure to ensure that the quality issue of the wall doesn't affect them.

A construction project produces different data from its different activities and rely on different data repositories or reference data like the as-designed BIM. Without a digital twin all this information would be kept in silos preventing any global view of the construction project. Thanks to the digital twin and its graph structure, it is possible to build a digital twin linking the different silos. The digital twin then becomes a shareable knowledge base between the different stakeholders of the construction project. A fine-grained access security model is provided to ensure each actor can control which data is shared with whom. It supports access rights for graph objects (nodes), relations and properties either node properties or relation properties.

An API is needed so that each stakeholder (manager, contractor, etc.) can both exploit (read) the digital twin knowledge and update it to ensure the knowledge is kept up to date among the construction project ecosystem. Figure 3 illustrates the benefits brought by sharing the knowledge in a digital twin:

- Bring a holistic view of the building construction project to improve its management.
- Shape a building construction view relevant for each team and job but benefiting from information from other teams (propagation of information and knowledge).
- Share the building project information and resources to avoid waste and improve efficiency.
- Federate the different technical systems used in the building construction project to hide the heterogeneity and provide a single point of entry.

Figure 3: Optimize construction management and production control with a Digital Building Twin Platform

The API defined in this document relies on the BIM2TWIN ontology [1] and the digital twin data organization in three layers [5]: the Raw Data Layer, the Logical Data Layer and the KPI Layer. The following section describes how this organization is reflected within the digital twin graph.

3 MAPPING OF THE B2T DATA LAYERS IN THE B2T GRAPH

B2T includes different Application WPs, each with its own requirements. The developed solutions within each WP need to be integrated on different levels. To this end, we propose multiple data layers that facilitate the integration and interaction between other systems.

The approach proposed by B2T project relies on the refinement of raw data produced in the construction project in information at the next refinement level, then knowledge to allow decision making.

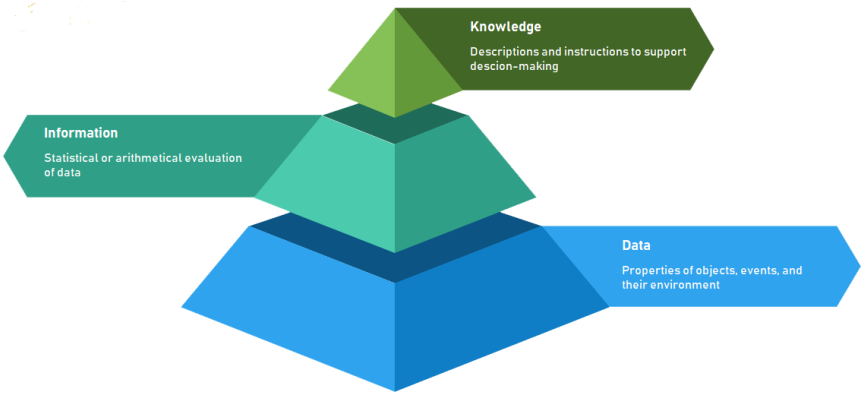

Figure 4: Data refinement process

As defined in [5], B2T has identified three layers with between them data flows to allow this refinement:

- The KPI Layer provides the project high-level indicators for a synthetic view of the project, the KPIs aiming at supporting decision making in the project. The KPIs are provided to the users through mainly the B2T Dashboard.
- The Logical Data Layer embeds atomic instances that mirror physical objects, processes and persons. The level of detail of this layer is a trade-off driven by KPIs. Data from the Raw Data Layer are processed, filtered, and organized in the logical model.
- The Raw Data Layer is the data lake of the digital twin. It stores data coming from automated (sensors) and manual data flow (manual entries) as they are produced.

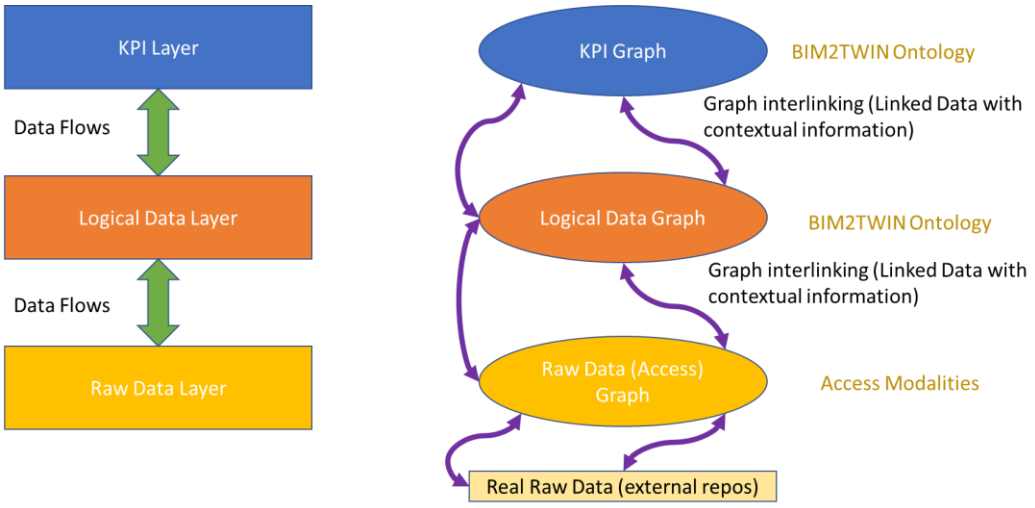

Figure 5: Mapping of the B2T Data Layers in the B2T Graph

Figure 5 describes how the B2T graph is mapped on the proposed data layer organization:

- The KPI Graph relies on the B2T Ontology to provide the expected high-level indicators and the graph relations provide access to the logical data related to a KPI. More than 20 KPIs are defined in [9] for Execution Excellence, Quality and Safety. Examples of KPIs are “Work in progress” (Process efficiency), “Order fulfilment quality” (Supply chain reliability), “Workforce capacity” (Resource efficiency), “Zero defect execution” (Quality assurance & control), “Zero defect work area” (Construction process quality), “Number of defects fixed” (Quality assurance efficiency) and “Site safety level” (Site safety).
- The Logical Data Graph relies also on the B2T Ontology to model the construction project with an increased level of details both for as-planned/as-performed and as-designed/as-built. Examples of information available are those related to physical objects (Building, Storey, Building Elements, Zone, Working Zone, Defect, etc.), persons and resources (Workers, Material, Equipment, etc.) and processes (Construction Schedule, Work Package? Resource Assignment, Inspection, Simulation, etc.).
- The Raw Data Graph provides a way to access the raw data from the graph to external repositories. Examples of raw data are point clouds provided by laser scanning performed for progress monitoring, images acquired at different wavelengths (visible, IR, multi-spectral range) for quality control, detailed planification data for resource optimization and project planification, sensors data flow. The Raw Data are not aimed to be duplicated in the graph which provides only the access modalities to the Raw Data (i.e., a description of the technical means to access the Raw Data such as an URL).

If the graph can be used to access the raw data, it can also be used to get an overview of all the data and then filter out the part that is relevant for a specific application.

The graph approach brings the benefit to describe with formal and semantic relations the links which exist between the different layers. The graph is also a convenient way to share the knowledge between the different project stakeholders and shape a view of the graph which is relevant for each business. The B2T graph is exploited by the APIs to achieve the refinement of information, inject new information and knowledge in the graph and enable the sharing of knowledge for a better decision-making.

4 DESCRIPTION OF APIS

This section includes the technical specification of different APIs that are essential for enabling the integration and interacting between different systems. In order to build a cohesive system, the API specification has been elaborated with the collaboration of the different WPs. Details on the requirements derived from these domains are in the scope of WP1 deliverables [6][7][8][9].

Table 1 and Table 2 represent the required API methods that enable progress monitoring and quality control that are mainly for WP3 and WP4. Table 3 represents the required API methods that enable safety management that are mainly for WP5. Table 4 represents the required API methods that enable equipment optimization that are mainly for WP6. Table 5 represents the required API methods that enable management of project planification that are mainly for WP7. Table 6 represents the required API methods that enable KPIs management and Dashboard features. Table 7 represents the required API methods for common needs on project data management. The following describes briefly each column name:

- The name of the API method.
- The definition of expected parameters whether mandatory or optional.
- The definition of expected response, which can be None (no response).
- The level of Importance to have this API method. Three levels are defined:
    - Mandatory: fully required with well-defined needs.
    - Secondary: mostly required but not yet well-defined needs.
    - Nice to Have: other API requirement neither Mandatory nor Secondary.
- The synchronous or asynchronous behaviour of the API method.
- The usage description of the API method.

4.1 Progress Monitoring and Quality Control

Progress monitoring in the construction phase verifies that the work is being built in accordance with the project, both with the required qualities, execution times, as well as with the functional specifications of the installations.

Quality Control on site could be defined as the technical verification (both of the materials and their execution) that the work has the specific technical characteristics necessary to avoid future failures, bad methods and deficient construction practices, and thus ensure the quality of the final product.

Table 1 – Progress Monitoring and Quality Control APIs

API Method Name	Input Parameters	Response	Requirement Level	Synchronous	Usage description
setAsDesignedGeometricElement	element_id: object_id e.g., wall id geometric_prob: geometric properties of element_id	Return successfully updated message	Mandatory	No	Adding the As-designed element's geometric properties/status for enabling the measurement the discrepancies between As-planned and As-performed for that element.
setAsPerformedGeometricElement	element_id: object_id e.g., wall id geometric_prob: geometric properties of element_id	Return successfully updated message	Mandatory	No	Adding the As-performed element's geometric properties/status for enabling the measurement the discrepancies between As-planned and As-performed for that element.
getAsDesignedGeometricElement	element: object_id e.g., wall id	Return geometric properties of a given element_id	Mandatory	No	Retrieving the As-designed element's geometric properties/status for measuring the discrepancies between As-planned and As-performed for that element.
getAsPerformedGeometricElement	element: object_id e.g., wall id	Return geometric properties of a given element_id	Mandatory	No	Retrieving the As-performed element's geometric properties/status for measuring the discrepancies between As-performed and As-designed for that element.
getAdjacentElements	element: object_id e.g., wall id	Return a list of adjacent elements given a particular element_id.	Mandatory	No	To highlight how quality issues related to one element (e.g., discrepancies between As-planned and As-designed/performed) affect or cascade to other elements. This also covers how quality issues cascade to buildings' elements.
updateAsDesignedGeometricElement	element: object_id e.g., wall id geometric_prob: geometric properties of element_id	Return successfully updated message	Mandatory	No	For updating geometric properties of building element for each scan or while monitoring the progress of the element.

API Method Name	Input Parameters	Response	Requirement Level	Synchronous	Usage description
updateAsPerformedGeometricElement	element: object_id e.g., wall id geometric_prob: geometric properties of element_id	Return successfully updated message	Mandatory	No	For updating geometric properties of building element for each scan or while monitoring the progress of the element.
getCountBuiltElementForBuilding	element_type: type of building's element e.g., wall storey_id: the story id/number	Return count (list of element_ids whose have updated geometry and marked as completed e.g. all ids of wall (i.e., element_type="wall") at a particular storey i.e., storey_id)	Mandatory	No	This will help to measure progress per system component e.g., the number of built walls compared with the number of planned walls.
getCountPlannedElementForBuilding	element_type: type of building's element e.g., wall storey_id: the story id/number	Return count of (list of expected element_ids) e.g. all ids of wall that don't have any geometry updated (i.e., element_type="wall") at a particular storey i.e., storey_id	Mandatory	No	This will help to measure progress per system component e.g., the number of built walls compared with the number of planned walls.
getElementStatus	element_id: the id of an element e.g., wall_id	Return the status e.g., completed of the element_id e.g., id of a wall	Nice to have	No	This will help later to measure progress per system component e.g., the number of (completed) built walls compared with the number of planned walls.
getElementInformation	element_type: type of building's element e.g., wall type_info: type of information e.g., material	Return relevant information for a given element	Nice to have	No	Obtain information about particular geometric element e.g., wall material related to other WPs. Perhaps specify enum in the type_of_info where internally we can call other methods required for WP4.
getQualityElementStatus	element_type: type of building's element e.g., wall quality_type: either geometric or texture i.e., WP3/WP4	Return element's quality status e.g., great, medium, low	Nice to have	No	Measure the quality of particular element from geometric i.e., WP3 or texture/surface i.e., WP4 perspectives. For instance, axis-deviation and WP3 can provide the ranger per level e.g., range of low quality, medium, and high.

Table 2 – Quality Control APIs

API Method Name	Input Parameters	Response	Requirement Level	Synchronous	Usage description
getAsDesignedElementGeometry	Possibility to return all building elements together with their subelements, filter by storey, or just one element by ID	As-designed building elements with their subelements, their geometry and their attributes	Mandatory	Yes	Required to compare the as-designed surfaces with the as-performed surfaces. Geometric information is needed to identify fissure/defect location in the BIM model based on defects identified on the construction site.
getElementsWithDefects	Possibility to return all building elements, filter by storey, or just one element by ID	All as-built elements that have a defect associated to them	Mandatory	Yes	During inspections, possible defects will be identified that are added to the as-built elements. Some defects might need additional analysis. By having information about all defected elements it can be evaluated which defects need further attention.
setDefect	Instance of defect class with all related attributes, its location, and the ID of the element to which it needs to be connected	Success code	Mandatory	No	One of the main outputs of WP4 is the information about detected surface defects. This information needs to be pushed to the Thing'in platform to be displayed in the dashboard and serves as input for other WPs.
setTolerance	Tolerance instance with all of its attributes and the element type(s) to which it should be applied to	Success code	Mandatory	No	As a starting point for every pilot site tolerances will be selected against which detected discrepancies will be compared. The tolerances need to be added to the Thing'in graph to make it clear which surface has to be checked according to which tolerance.

API Method Name	Input Parameters	Response	Requirement Level	Synchronous	Usage description
getTolerance	Building element specified by ID	All tolerances that are associated to a building element	Mandatory	Yes	Once a discrepancy is detected, it is necessary to know which tolerances need to be applied. Therefore, all tolerances related to a specific building element need to be known.
setImage	Camera image with information about which camera captured it and at which point in time	Success code	Mandatory	No	Images of building elements (with defects) will be made. These images should be pushed to the central platform so that, e.g., a defect can be associated with an image on which it can be seen, or also that other WPs can use the images to interpret them according to their goals.
setDepthMap	Depth map (probably only the z value) together with the meta data (timestamp and device that captured it)	Success code	Secondary	No	The depth map is obtained from a depth sensor like LIDAR or TOF sensors. The depth map can be used to match the crack/defect location on the framed surface with respect to the BIM. This information can also be relevant for other WPs.
getQRCodeLocation	QRCode ID	Location of QRCode	Secondary	Yes	Some QRCodes will be placed on building elements to have a reference point on images with known position. QRCodes will show up on defect images. It is necessary to get information about the exact location of a specific QRCode. (Maybe the location information can be directly encrypted into the QRCode.)
updateElementStatus	Element ID and new status	Success code	Secondary	No	One of the use cases is dedicated to detect the current status of a concrete column. Once a status change has been detected this information needs to be updated on the central platform. (Maybe even the related processes need to be updated but this is still up for discussion.)

4.2 Safety Management

The main objective of Safety Management is to ensure the occupational health and safety (OHS) of construction workers through a system of prevention through design (PtD) in construction operation design and planning, proactive real-time risk detection and warning on construction sites, and rapid personalized feedback in learning and decision making.

Table 3 – Safety Management APIs

API Method Name	Input Parameters	Response	Requirement Level	Synchronous	Usage description
getAsDesigned	None	Geometrical representations of construction elements and attributes	Mandatory	No	Construction 3D model
getAsPlanned	date interval	Construction schedule formulated as activities	Mandatory	No	The date interval should support inclusion and exclusion
getAsBuild	None	Current element status overview	Mandatory	No	The current status of all construction elements
getActiveWorkZones	date interval	Work zones that overlap with the date interval	Mandatory	No	The date interval should support inclusion and exclusion
getWorkZones	date interval	Work zones that overlap with the date interval	Mandatory	No	The date interval should support inclusion and exclusion
getSafetyPerformance	None	List of all crews with their safety performance described as a list of events and the probability	Mandatory	No	The safety performance recording is used to calculate the risk levels of extracted hazards
addHazardSpace	WKT representation, ID, type, risk level, Start time, end time, Source process	Regular API response codes	Mandatory	No	Add a spatial zone, in which a hazard of type may occur. The hazard will be connected to the source, and the subject are overlapping work zones/work crews, within the time interval between start and end

API Method Name	Input Parameters	Response	Requirement Level	Synchronous	Usage description
addMitigationMeasure	Equipment type, placement, element	Regular API response codes	Mandatory	No	Adds a temporary equipment to mitigate a hazard, that has been found during the analysis. This could include safety guard railing or safety nets
addSafetyRelatedTask	Task type, contractor	Regular API response codes	Mandatory	No	Add a task e.g., for installation, removal or inspection of safety mitigation equipment
addWorkerObservation	Crew, workzone,location	Regular API response codes	Mandatory	No	Add an observation of a worker in location (x, y, z) at time t
addIncidentObservation	Incident Description, Time, location	Regular API response codes	Mandatory	No	Adds an observation of an incidents extracted from data capture and processing

4.3 Equipment Optimization

The main objective of Equipment Optimization is to provide an efficient control of equipment on the construction site, based on the capture and the analysis of its current position, its current activity and its current condition.

Table 4 – Equipment Optimization APIs

API Method Name	Input Parameters	Response	Requirement Level	Synchronous	Usage description
getModel	LOD: Specifies the required LOD [architectural design LOD 400, structural engineering LOD 400, construction site layout LOD 200].	Entire model is returned (geometry plus attribute information)	Mandatory	No	Retrieval of the model for local visualization of the detected equipment. The model should contain static objects (e.g., walls, columns, containers, ...). If LOD is not available or possible, then the existing model should simply be reverted.
getWorkingZones	No parameters	Returns a list of working zones that are planned and visible in the model.	Mandatory	No	The working zones should be transmitted in such a way that it is easy to find the zones in the model. Perhaps each object in the model should be associated with a working zone ID.
setEquipmentPosition	Position: Position of the detected equipment (x,y,z). EquipmentID: ID of the detected equipment. Will be a string. AdditionalInformation: Information that is passed as a key value pair (for example, measured speed or the direction of movement of a machine). The values can be passed as a string for simplification and unification.	Boolean Value, to check if the platform has processed the data.	Mandatory	Yes	This method passes the position of the detected equipment and additional extracted information, such as speed, direction, size or orientation, to the platform.
getEquipmentPosition	EquipmentID: ID of the equipment (String).	Returns position and all other stored information about the equipment.	Mandatory	Yes	Query information already stored about an equipment.

API Method Name	Input Parameters	Response	Requirement Level	Synchronous	Usage description
getEquipmentWorkingZones	EquipmentID: ID of the equipment.	Returns all planned working zones for the equipment. If available, planned working times for each working zone.	Mandatory	No	This function returns the planned working zones where the equipment will be used.
getEquipment	EquipmentName: If specified, search only for equipment that matches the name (e. g. Excavator). Otherwise, all known equipment should be returned.	Returns the list of known equipment.	Mandatory	No	This method returns the list of known equipment. Additional information (e.g., ID, name, type, size, Position, Link to activity list [activity list], ...) available for the equipment should also be returned as a list.
getLastModelUpdate	None	Time stamp (date, time))	Mandatory	No	It should simply return the last known update of the model.
getAllEquipmentInWorkingZone	workingZoneID: ID of the working zone	All equipment should be returned that is in the Working Zone	Mandatory	No	By passing the WorkingZoneID as a parameter, all known equipment in the working zone should be searched for and returned.
getModelPart	workingZoneID: Id of the working zone that is needed. Can be a string or a numeric number. LOD: Specifies the required LOD [architectural design LOD 400, structural engineering LOD 400, construction site layout LOD 200].	Part of the model is returned (geometry and attribute information)	Secondary	No	Retrieval of the partial model for local visualization of the detected devices. The model should contain static objects (e.g., walls, columns, containers, ...). This function should only return something if the model is divided into workspaces. If LOD is not available or possible, then the existing model should simply be reverted.

API Method Name	Input Parameters	Response	Requirement Level	Synchronous	Usage description
getConstructionActivities	workingZoneID: ID of the working zone for which construction activities are to be determined (optional). If no ID is specified, all working zones will be searched. EquipmentID: ID of the equipment for which construction activities are to be determined (optional). If no ID is specified, all equipment will be considered.	List of the planned activity to the activities used.	Nice to Have	No	This function should be used to query information about the planned activities that are related to the equipment.
setWorkingZoneMessage	Message: Message to be assigned to a Working Zone. EquipmentID: Linking the message to an EquipmentID. Duration: Duration of the existence of the message.	none	Nice to Have	No	This message could be an announcement of a work that people in the field should pay attention to. For example, a crane may announce a dangerous transport.
getWorkingZoneMessages	workingZoneID: ID of the working zone	Messages that exist in the Working Zone together with the Equipment ID	Nice to Have	No	Query of messages in the work zones

4.4 Project Planification

The main objective of Project Planification is the optimization of construction planning. It will do this by predicting possible outcomes of changes to design or plan, using simulations based on the situational awareness provided by the digital building twin.

Table 5 – Project Planification APIs

API Method Name	Input Parameters	Response	Requirement Level	Synchronous	Usage description
getAsPerformedWorkStatus	date_range: optional - filter to retrieve only status within a specific range, default is to retrieve the full status	full graph of relevant as performed construction, operation, and action in graph or tabular format	Mandatory	Yes	Retrieving of the as performed work status for adjusting the baseline production plan
getBaselinePlan	date_range: optional - filter to retrieve only status within a specific range, default is to retrieve the full status	full graph of the as-planned BIM2TWIN core graph between the relevant task date range	Mandatory	Yes	Retrieving the baseline plan to generate alternative plans and to parametrically generate the simulation model
getAsPlannedLocationBreakdownStructure		full as planned space breakdown structure in graph or tabular format	Mandatory	Yes	Retrieving the location breakdown structure to generate alternative plans and to parametrically generate the simulation model
getBuildingGeometry		exact geometry of all as-designed building elements	Secondary	Yes	For visualization of simulation result through Unity rendering engine
getBoundingBox		bounding box of all as-designed building elements	Secondary	Yes	For generating representations within the simulation model
storeSimulationOutput	construction_plan_id: mandatory - the relevant baseline or alternative plan that the simulation output is linked to output_data: mandatory - file containing the simulation output data		Secondary	No	Links a simulation output data file with the associated construction plan in the database as a blob

API Method Name	Input Parameters	Response	Requirement Level	Synchronous	Usage description
storeConstructionPlan	associated_baseline_plan_id: mandatory - the id of the baseline plan stored in the DBTP database that the plan is associated with construction_plan: mandatory - a complete construction plan according to the BIM2TWIN core ontology is_new_baseline: mandatory	plan_id: mandatory - the id of the stored plan within the Thing'in database	Secondary	No	Storing a proposed construction plan into the graph database. We imagine that to achieve this function would **entail a set of "sub-APIs"** that creates new instances of graph elements in the database. If *is_new_baseline* is *true*, the input plan will be labeled in the Thing'in database as the new BP, and the old BP is archived
storeKPI	kpi_class: mandatory - the input KPI's corresponding B2T ontology class associated_plan_id: mandatory - the id of the associated plan within the Thing'in database kpi_value: mandatory - value of the KPI kpi_unit: mandatory - unit of the KPI value		Secondary	No	Storing a calculated KPI into the Thing'in database with link to the associated construction plan
getAsPerformedLocationBreakdownStructure		full as performed space breakdown structure in graph or tabular format	Nice to have	Yes	Retrieving the location breakdown structure to generate alternative plans and to parametrically generate the simulation model
getCrewProductivity	CrewID: mandatory date_range: optional - range to calculate the crew productivity, default is to sample all available data points	crew_productivity - probability distribution of crew productivity based on sampling from the selected date range	Nice to have	Yes	Calculate the probability distribution of crew productivity based on recorded project status information

4.5 KPIs Management (Project Dashboard)

The main objective of KPIs management is to provide the calculation of the KPIs and information that aims to be displayed in the synthetized project view provided by the B2T dashboard.

Table 6 – KPIs Management APIs

API Method Name	Input Parameters	Response	Requirement Level	Synchronous	Usage description
getTasks	intervalStartDate, intervalEndDate, task type (optional), resource type (optional), equipment type (optional), working zone (optional), finished only (optional), unfinished only (optional), delayed only (optional)	list of tasks with id, duration, real start date, real end date, expected end date, isDelayed, task type, resource type, quantity of resource consumed, equipment type, working zone, number of defects (on related elements)	Mandatory	Yes	Needed to calculate: Average duration of task, Average consumption of resource by task, Number of unfinished task, Average duration of task by working zone, Average consumption of resource by task type, Number of zero defect tasks, Ratio of finished tasks, Average number of defects per tasks, Number of defects per week
getWorkingZones		list of contruction site working zones with id name, zone type	Mandatory	Yes	Needed to allow user to filter results
getElementTypes		list element types with id and label	Mandatory	Yes	Needed to allow user to filter results
getResourceTypes		list resource types with id, label and unit	Mandatory	Yes	Needed to allow user to filter results
getEquipementTypes		list equipement types with id and label	Mandatory	Yes	Needed to allow user to filter results
getElements	intervalStartDate, intervalEndDate working zone (optional), element type (optional)	list of as-designed element (compared to as-build status) with id, element type, working zone, isFinished, total number of defects, number of defect solves, isDelayed	Mandatory	Yes	Needed to display 3D view Needed to calculate: Number of defects solved per week
getKpis		list of KPIs with id, label, description, units	Mandatory	Yes	Needed to display list of available KPIs to the user

API Method Name	Input Parameters	Response	Requirement Level	Synchronous	Usage description
getOrders	intervalStartDate, intervalEndDate, resource type (optional), finished only (optional)	list of orders with id, resource type, real end date, expected end date	Mandatory	Yes	Needed to calculate: Ratio of order performed on time Ratio of order delayed
getWorkers	intervalStartDate (optional), intervalEndDate (optional), working zone (optional)	Actual list of workers with id, assigned task (null if not assigned), working zone, work duration (in jours)	Mandatory	Yes	Needed to calculate: Ratio of workers assigned
getEquipements	intervalStartDate, intervalEndDate, equipment type (optional), working zone (optional)	list of equipements with id, working zone, assigned task	Mandatory	Yes	Needed to calculate: Ratio of equipement used
getWorkpackages	intervalStartDate, intervalEndDate, working zone (optional), only with defects (optional), only no defects (optional), only finished (optional), only unfinished (optional)	list of workpackages with id, workpackage type, real end date, expected end date, working area, isDelayed, number of defects (on related elements)	Mandatory	Yes	Needed to calculate: Ratio of zero-defect workpackages, Average delay per workpackage due to defect, Ratio of zero-defect workpackages, Average number of defects per workpackage
getObservations	intervalStartDate, intervalEndDate, observation type (optional), working zone (optional)	list of observation with id, observation type, target, date, working zone, isCorrect (observation status)	Mandatory	Yes	Needed to calculate: Ratio of correct observation
updateKpiRecord	record id, startDate,endDate, kpi id, sample quantity, reference quantity, kpi value, working zone (optional), task type (optional), resource type (optional), equipement type (optional)		Mandatory	Yes	Needed to materialize KPI calculation

API Method Name	Input Parameters	Response	Requirement Level	Synchronous	Usage description
getKpiRecord	intervalStartDate, intervalEndDate, task type (optional), resource type (optional), equipment type (optional), working zone (optional)	return list of KPI record with id, startDate, endDate, kpi id, sample quantity, reference quantity, kpi value, task type, resource type, equipment type, working zone	Mandatory	Yes	Needed to request KPI calculation
getGeometry	format = B3DM, buildingID	return the Building geometry as 3D tiles (B3DM standard https://docs.opengeospatial.org/cs/18-053r2/18-053r2.html)	Mandatory	Yes	Use for 3D viewer
getCycleTime	intervalStartDate, intervalEndDate, task type (optional), resource type (optional), equipment type (optional), working zone (optional)	return list of KPI record with id, startDate, endDate, kpi id, sample quantity, reference quantity, kpi value, task type, resource type, equipment type, working zone	Nice to have	no (can be materialized as kpi records)	Calculation (one record by week): The cycle time measures the time required to complete a cycle of a process for a specific component
getThroughput	intervalStartDate, intervalEndDate, task type (optional), resource type (optional), equipment type (optional), working zone (optional)	return list of KPI record with id, startDate, endDate, kpi id, sample quantity, reference quantity, kpi value, task type, resource type, equipment type, working zone	Nice to have	no (can be materialized as kpi records)	Calculation (one record by week): Measures the number of units produced by a specific manufacturing process during a specified period.

API Method Name	Input Parameters	Response	Requirement Level	Synchronous	Usage description
getWorkInProgress	intervalStartDate, intervalEndDate, task type (optional), resource type (optional), equipment type (optional), working zone (optional)	return list of KPI record with id, startDate, endDate, kpi id, sample quantity, reference quantity, kpi value, task type, resource type, equipment type, working zone	Nice to have	no (can be materialized as kpi records)	Calculation (one record by week): number of unfinished units in a specific moment.
getCycleTimeVariability	intervalStartDate, intervalEndDate, task type (optional), resource type (optional), equipment type (optional), working zone (optional)	return list of KPI record with id, startDate, endDate, kpi id, sample quantity, reference quantity, kpi value, task type, resource type, equipment type, working zone	Nice to have	no (can be materialized as kpi records)	Calculation (one record by week): identification of possible errors in production and enhancing continuous improvements and capabilities to perform root cause analysis.
getThroughputVariability	intervalStartDate, intervalEndDate, task type (optional), resource type (optional), equipment type (optional), working zone (optional)	return list of KPI record with id, startDate, endDate, kpi id, sample quantity, reference quantity, kpi value, task type, resource type, equipment type, working zone	Nice to have	no (can be materialized as kpi records)	Calculation (one record by week): Variability of throughput in a related unit of measure
getOrderFulfillmentTimeliness	intervalStartDate, intervalEndDate, task type (optional), resource type (optional), equipment type (optional), working zone (optional)	return list of KPI record with id, startDate, endDate, kpi id, sample quantity, reference quantity, kpi value, task type, resource type, equipment type, working zone	Nice to have	No (can be materialized as kpi records)	Calculation (one record by week): number of on-time deliveries out of total deliveries

API Method Name	Input Parameters	Response	Requirement Level	Synchronous	Usage description
getOrderFulfillmentQuality	intervalStartDate, intervalEndDate, task type (optional), resource type (optional), equipment type (optional), working zone (optional)	return list of KPI record with id, startDate, endDate, kpi id, sample quantity, reference quantity, kpi value, task type, resource type, equipment type, working zone	Nice to have	no (can be materialized as kpi records)	Calculation (one record by week): number of zero-defect deliveries out of total deliveries
getOrderFulfillmentCycleTime	intervalStartDate, intervalEndDate, task type (optional), resource type (optional), equipment type (optional), working zone (optional)	return list of KPI record with id, startDate, endDate, kpi id, sample quantity, reference quantity, kpi value, task type, resource type, equipment type, working zone	Nice to have	No (can be materialized as kpi records)	Calculation (one record by week): Cycle time from order to delivery
getWorkforceUtilizationRate	intervalStartDate, intervalEndDate, task type (optional), resource type (optional), equipment type (optional), working zone (optional)	return list of KPI record with id, startDate, endDate, kpi id, sample quantity, reference quantity, kpi value, task type, resource type, equipment type, working zone	Nice to have	No (can be materialized as kpi records)	Calculation (one record by week): workers able to perform value-adding work
getEquipmentUtilizationRate	intervalStartDate, intervalEndDate, task type (optional), resource type (optional), equipment type (optional), working zone (optional)	return list of KPI record with id, startDate, endDate, kpi id, sample quantity, reference quantity, kpi value, task type, resource type, equipment type, working zone	Nice to have	No (can be materialized as kpi records)	Calculation (one record by week): ratio of when the equipment was used compared to the total time on site

API Method Name	Input Parameters	Response	Requirement Level	Synchronous	Usage description
getThroughput	intervalStartDate, intervalEndDate, task type (optional), resource type (optional), equipment type (optional), working zone (optional)	return list of KPI record with id, startDate, endDate, kpi id, sample quantity, reference quantity, kpi value, task type, resource type, equipment type, working zone	Nice to have	No (can be materialized as kpi records)	Calculation (one record by week): percentual number of planned tasks fully completed in each period
getNumberOfDefectsPerWork	intervalStartDate, intervalEndDate, task type (optional), resource type (optional), equipment type (optional), working zone (optional)	return list of KPI record with id, startDate, endDate, kpi id, sample quantity, reference quantity, kpi value, task type, resource type, equipment type, working zone	Nice to have	No (can be materialized as kpi records)	Calculation (one record by week): number of defects in each work package
getZeroDefectExecutionsOutOfTotalExecutions	intervalStartDate, intervalEndDate, task type (optional), resource type (optional), equipment type (optional), working zone (optional)	return list of KPI record with id, startDate, endDate, kpi id, sample quantity, reference quantity, kpi value, task type, resource type, equipment type, working zone	Nice to have	No (can be materialized as kpi records)	Calculation (one record by week): frequency of the defect count per work package
getWorkPackageDelaysDueToUnfixedDefect	intervalStartDate, intervalEndDate, task type (optional), resource type (optional), equipment type (optional), working zone (optional)	return list of KPI record with id, startDate, endDate, kpi id, sample quantity, reference quantity, kpi value, task type, resource type, equipment type, working zone	Nice to have	No (can be materialized as kpi records)	Calculation (one record by week): Delay caused by issues reported on the execution of tasks.

API Method Name	Input Parameters	Response	Requirement Level	Synchronous	Usage description
getZeroDefectWork	intervalStartDate, intervalEndDate, task type (optional), resource type (optional), equipment type (optional), working zone (optional)	return list of KPI record with id, startDate, endDate, kpi id, sample quantity, reference quantity, kpi value, task type, resource type, equipment type, working zone	Nice to have	No (can be materialized as kpi records)	Calculation (one record by week): frequency of the zero-defect executions
getNumberOfDefectsPerWork	intervalStartDate, intervalEndDate, task type (optional), resource type (optional), equipment type (optional), working zone (optional)	return list of KPI record with id, startDate, endDate, kpi id, sample quantity, reference quantity, kpi value, task type, resource type, equipment type, working zone	Nice to have	No (can be materialized as kpi records)	Calculation (one record by week): Number of defects identified and fixed by the workgroup during their task execution out of the total amount of defects on a specific working area.
getSiteSafetyLevel	intervalStartDate, intervalEndDate, task type (optional), resource type (optional), equipment type (optional), working zone (optional)	return list of KPI record with id, startDate, endDate, kpi id, sample quantity, reference quantity, kpi value, task type, resource type, equipment type, working zone	Nice to have	No (can be materialized as kpi records)	Calculation (one record by week): frequency of correct safety observations out of total safety observations.
getDefectSolving	intervalStartDate, intervalEndDate, task type (optional), resource type (optional), equipment type (optional), working zone (optional)	return list of KPI record with id, startDate, endDate, kpi id, sample quantity, reference quantity, kpi value, task type, resource type, equipment type, working zone	Nice to have	No (can be materialized as kpi records)	Calculation (one record by week): number of defects solved in each period

API Method Name	Input Parameters	Response	Requirement Level	Synchronous	Usage description
getAccidentFrequence	intervalStartDate, intervalEndDate	Return list of KPI record with id, startDate, endDate, kpi id, sample quantity, reference quantity, kpi value, task type, resource type, equipment type, working zone	Nice to have	No (can be materialized as kpi records)	Calculation (one record by week): accident frequency is counted by dividing all the accidents that lead to the disablement of over one workday by one million working hours.

4.6 Project Management

The Project Management defined methods are required to keep in the platform the project intent information (as-built and as-planned) updated. For example, if the construction manager on a construction site (e.g., a B2T pilot site) decides to make a change to the construction schedule, then we also need to update the information in the digital twin graph accordingly. The same applies for other types of changes that the construction management decides on.

Table 7 – Project Management APIs

API Method Name	Input Parameters	Response	Requirement Level	Synchronous	Usage description
updateSchedule	Excel file of the new construction schedule	the current baseline plan will be changed to "not baseline" (not current version anymore) and the new schedule will be added (current baseline plan). Some connections from as-planned to as-preformed need to be copied from the old schedule to the new schedule	Mandatory	No	Every time the construction schedule is updated the information needs to be transferred to the Thing'in graph.
updateResourceAssignment	Excel file of the updated resource assignment	Replace parts of the resource assignment of the current baseline construction schedule	Mandatory	No	Every time the resource assignment is updated the information needs to be transferred to the Thing'in graph

5 ADDITIONAL INFORMATION ON DEVELOPING WITH THE DBTP

In this section we remind the other tools provided by the platform to ease development of business applications for building construction project.

Additional details are available in the specification of the DBTP [4].

5.1 IFC injector

The main goal of this injector is to parse an IFC (Industry Foundation Classes) input file, extract relevant topologic data and create in the DBTP a digital representation based on that information.

The IFC injector maps the IFC elements onto the B2T ontology [1], more specifically onto the BOT ontology [2] which is part of the B2T ontology. The injected IFC elements are then directly usable through the B2T API.

5.2 Data (other than IFC) injection

IFC is an important set of data to feed the B2T Digital Twin but other set of data are needed like project planification data, quality control data, etc. The graph-based platform provides different means for a developer to inject the data and create and feed a digital twin:

- Generic Data injector for set of data in CSV or JSON format.
- Batch data injection API to inject data in the graph in an asynchronous way.
- BLOB (Binary Large Object) API to link to the graph arbitrary content files.

6 CONCLUSION

In its first version, this document aims to provide a specification of the API of the graph-based Digital Building Twin Platform for BIM2TWIN project.

The API is a key furniture of the project, describing how the different business of a construction project will exploit the digital twin, feed it and enable the knowledge sharing among the construction project stakeholders and ease the decision-making.

Additional iterations of the document should occur to refine the API description, clean it from redundancies and consolidate if “secondary” or “nice to have” API definition should be integrated in a finalized version. The lessons learned from the development of the demonstrators will also help a lot in strengthen the API proposal.

Eventually, the dissemination of this specification should be studied in the activities of WP9 (Dissemination, standardization, exploitation).

7 APPENDIX A: LITERATURE

7.1 References

[1] The BIM2TWIN Ontology: https://github.com/jschlenger/BIM2TWIN-Ontologies/

[2] The BOT ontology: https://w3c-lbd-cg.github.io/bot/

[3] Thing in the future, a graph-based Digital Twin platform: https://www.thinginthefuture.com, https://thinginthefuture.bim2twin.eu/login, https://wiki.thinginthefuture.com/

[4] Specification of the graph-based platform for managing the digital twin - BIM2TWIN D2.6 deliverable

[5] Digital Building Twin Requirements Analysis and Data Model – BIM2TWIN D2.1 deliverable

[6] As-is practices analysis and end-user requirements – BIM2TWIN D1.1 deliverable

[7] Definition of the digital workflows for the construction process – BIM2TWIN D1.2 deliverable

[8] KPIs for evaluating theconstruction process behaviour – BIM2TWIN 1.3 deliverable

[9] Digital Twin Dashboard Requirements & Specifications – BIM2TWIN 1.4 deliverable

[10] A. Abbas, G. Privat: Bridging Property Graphs and RDF for IoT Information Management. *International Semantic Web Conference* (ISWC 2018), Workshop on Scalable Semantic Web Knowledge Base Systems, Monterey, California, USA; 10/2018

[11] Dana Popovici, Gilles Privat, Capturing the Structure of Internet of Things Systems with Graph Databases for Open Bidirectional Multiscale Data Mediation. *The Second International Workshop on Large-scale Graph Storage and Management*, Rome; 05/2015

[12] G. Privat, A. Abbas: Cyber-Physical Graphs vs.RDF graphs, *W3C Workshop on Web Standardization for Graph Data*, Berlin, 03/2019

[13] Wenbin Li, Gilles Privat, José Manuel Cantera, Martin Bauer, Franck Le Gall: Graph-based Semantic Evolution for Context Information Management Platforms. 2018 *Global Internet of Things Summit* (GIoTS), Bilbao, Spain; 06/2018, DOI:10.1109/GIOTS.2018.8534538