

PROJECT REPORT

SmartWater

Submitted by

Anup Kumar Duya

2111981048

Supervised By

Dr. Nagesh Kumar



Department of Computer Science and Engineering

CHITKARA UNIVERSITY

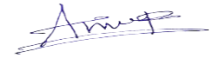
CONTENTS

Title

1. Declaration
2. Acknowledgement
3. Introduction
4. Abstract
5. Methodology
6. Tools and Technologies
7. Implementation
8. ITK Code Implementation/Result/Explanation
9. Conclusion

DECLARATION

We here by declare that the project work titled, SmartWater submitted as part of Bachelor's degree in CSE, at Chitkara University, Himachal Pradesh, is an authentic record of our own work carried out under the supervision of Mr. Nagesh Kumar.



Signature
Anup Kumar Duya

ACKNOWLEDGEMENT

I express my sincere gratitude to everyone who contributed to the successful completion of this project on the SmartWater in the context of PLM Teamcenter. I extend my heartfelt thanks to my project guide and mentors for their invaluable guidance, encouragement, and constructive feedback throughout the project. Their expertise and insights were instrumental in understanding ITK and its application within Teamcenter.

I am also deeply thankful to my institution, Chitkara University, for providing me with the resources and a conducive environment to undertake this project. Special thanks to my faculty members and peers for their support and collaboration.

Lastly, I would like to acknowledge the developers and documentation contributors of Teamcenter ITK, whose work served as a strong foundation for this project. Their contributions greatly facilitated the development of practical.

Introduction

What is PLM?

Product Lifecycle Management (PLM) is a systematic approach to managing the complete lifecycle of a product, from its inception, design, and development to production, usage, and eventual disposal. PLM integrates people, processes, and data across various stages of the product lifecycle to enhance collaboration, improve efficiency, and ensure data consistency.

It serves as a central repository for product-related information, enabling seamless communication among stakeholders, including designers, engineers, manufacturers, and customers. PLM solutions help organizations streamline workflows, reduce time-to-market, and maintain compliance with industry standards. In industries like automotive, aerospace, and manufacturing, PLM systems like Teamcenter play a vital role in fostering innovation, improving product quality, and achieving cost savings by ensuring better decision-making throughout the product lifecycle.

PLM is a cornerstone technology for modern industries, enabling organizations to manage the complexities of product development in an increasingly competitive and innovation-driven environment. By bridging the gap between diverse teams such as engineering, manufacturing, and supply chain, PLM fosters an integrated approach to product creation and lifecycle management. It ensures that all stakeholders have access to accurate, up-to-date information, which is critical for making informed decisions at every stage of a product's journey.

A key aspect of PLM is its ability to handle vast amounts of data generated throughout the lifecycle of a product, from 3D models and simulations to maintenance records and customer feedback. This centralized data not only helps to reduce redundancy and errors but also supports regulatory compliance by maintaining traceability and audit readiness. Furthermore, PLM systems provide tools to analyze performance metrics, enabling companies to identify trends and optimize their processes for future product development. This makes PLM an essential enabler of continuous improvement and long-term sustainability in product innovation.

Abstract

The development of a "smart water" soft drink, which integrates advanced ingredients, functional plant additives, and innovative packaging to enhance hydration and health benefits, can greatly benefit from the capabilities of Teamcenter Product Lifecycle Management (PLM). By utilizing Teamcenter, manufacturers can efficiently manage the entire lifecycle of the product, from concept development and formulation to production and marketing. Teamcenter offers a centralized platform for collaboration, enabling cross-functional teams such as R&D, production, and marketing to work together seamlessly, ensuring consistency in product quality, regulatory compliance, and timely market delivery. Key advantages include improved product data management, streamlined change management processes, and enhanced traceability of ingredients and formulations. Furthermore, Teamcenter facilitates the integration of various tools and systems for faster prototyping, testing, and iteration, while allowing for detailed tracking of production processes and packaging innovation. The use of Teamcenter PLM for the development of smart water soft drinks results in reduced time-to-market, better product quality control, and the ability to respond swiftly to market demands, leading to a competitive edge in the beverage industry. Ultimately, it enables the efficient creation of healthier, functional, and sustainable beverage products that meet consumer preferences in a rapidly evolving market.

Tools and Technologies:

There are a number of tools and technologies which are used in this project. Some of the tools and technologies used are as follows:

Teamcenter: Product Lifecycle Management (PLM) software developed by Siemens Digital Industries. Teamcenter is a widely used Software. PLM systems like Teamcenter are designed to manage product-related information throughout its lifecycle, from initial design and development through manufacturing, maintenance, and disposal.

BMIDE: BMIDE, which stands for Business Modeler Integrated Development Environment, is a critical component of Teamcenter, Siemens' Product Lifecycle Management (PLM) solution. The BMIDE tool is used for configuring and customizing the data model within Teamcenter. It allows organizations to adapt Teamcenter to their specific needs and business processes by defining and managing data model elements, business rules, and other configuration settings. BMIDE is a powerful tool that empowers organizations to tailor Teamcenter to their specific requirements, ensuring that the PLM system aligns with their business processes and supports their unique workflows. It plays a crucial role in the adaptability and flexibility of Teamcenter as a comprehensive PLM solution.

AWC: Active Workspace (AWC) is the modern web-based user interface for Teamcenter, Siemens' Product Lifecycle Management (PLM) software. AWC provides users with an intuitive and unified environment for accessing, visualizing, and interacting with PLM data and processes. It is designed to enhance collaboration, streamline workflows, and provide a more user-friendly experience across various devices. Active Workspace is designed to provide a modern and streamlined user experience for Teamcenter users, facilitating collaboration, improving accessibility, and enhancing overall productivity in PLM processes.

ITK APIs: ITK (Integration Toolkit) APIs are a set of C-based programming interfaces provided by Teamcenter to enable advanced customization and server-side automation. These APIs offer direct access to Teamcenter's core functionalities, such as querying, creating, and modifying business objects, as well as enforcing business rules. ITK APIs allow developers to implement custom logic tailored to an organization's workflows. They ensure robust, scalable, and secure interactions with the Teamcenter server. By leveraging ITK APIs, developers can extend the out-of-the-box capabilities of Teamcenter, integrate third-party systems, and achieve seamless process automation.

SQL: In the context of Teamcenter, SQL (Structured Query Language) is often used for querying and accessing data stored within the Teamcenter database. Teamcenter is a comprehensive Product Lifecycle Management (PLM) system developed by Siemens, and it relies on a relational database to store and manage product-related information. SQL is employed to retrieve, update, and manipulate this data. It's important to note that while SQL can be a powerful tool for interacting with the Teamcenter database, direct manipulation of the database should be approached with caution. Teamcenter provides a set of APIs (Application Programming Interfaces) and tools, including ITK (Integration Toolkit) and SOA (Service-Oriented Architecture), which are often preferred for interacting with the system programmatically.

Visual studio Community: Visual Studio Community is a free Integrated Development Environment (IDE) provided by Microsoft, designed for coding, debugging, and compiling applications in various programming languages, including C/C++. It is a primary tool for ITK development, offering features like IntelliSense, code navigation, and integrated debugging. For ITK, developers use Visual Studio to write server-side customizations, link libraries, and compile source code into deployable binaries. Its user-friendly interface and powerful tools enable efficient development of ITK projects, making it an essential choice for professionals working with Teamcenter customizations.

Batch Utility: Batch Utility in Teamcenter is a tool designed for automating bulk operations or repetitive tasks, such as importing data, mass updating attributes, or exporting information. It helps organizations save

time and reduce errors by allowing predefined operations to be executed programmatically. Batch Utility is often used in combination with ITK customizations to handle large datasets efficiently. It provides flexibility and scalability, making it an essential tool for organizations dealing with significant volumes of product data or requiring frequent updates to their Teamcenter environment.

Rule Handler: A Rule Handler in Teamcenter is a server-side customization that defines and enforces business logic or validation rules on objects or workflows. It ensures that operations meet predefined conditions, such as data validation or compliance requirements, before being executed. Rule Handlers are implemented using ITK and are triggered during specific events, such as object creation or modification. They help maintain data consistency and enforce business policies across the system, ensuring that Teamcenter operates within the defined organizational guidelines.

Action Handler: An Action Handler in Teamcenter is a piece of custom logic that performs specific actions when triggered by events or workflows. Action Handlers are used to automate tasks such as updating attributes, sending notifications, or integrating with external systems. They are implemented using ITK APIs and can be configured to run during various operations within Teamcenter. Action Handlers enhance productivity by automating routine tasks and ensuring that processes follow the organization's predefined workflows, reducing manual effort and errors.

Pre Condition: Pre-Conditions in Teamcenter are logical checks or validations applied before an operation is executed. They are used to ensure that specific criteria are met, such as user permissions, data integrity, or object status, before allowing the operation to proceed. Pre-Conditions are typically implemented using ITK or configured within BMIDE. By enforcing these validations, Pre-Conditions help maintain system integrity, prevent unauthorized actions, and ensure compliance with business rules, making them a critical component of Teamcenter workflows.

Post Action: Post-Actions in Teamcenter are operations or tasks executed after a primary operation has been completed successfully. They are used to automate follow-up actions, such as logging events, updating related objects, or triggering notifications. Post-Actions are implemented using ITK APIs and are commonly used in workflows to ensure that subsequent steps are handled automatically. By automating post-operation tasks, Post-Actions enhance efficiency and help maintain consistency across processes, ensuring smooth transitions within the Teamcenter environment.

Methodology

What is SmartWater?

Smart water is an innovative soft drink designed to enhance hydration and provide additional health benefits through the integration of advanced ingredients, functional additives, and cutting-edge packaging. It goes beyond traditional water or beverages by offering features such as improved electrolyte balance, added vitamins, and enhanced bioavailability for better absorption.

SmartWater plans to innovate its SmartWater product line by introducing a new flavor using plant-based ingredients which offer several key benefits:

- 1. Healthier Alternative:** Plant-based ingredients often contain natural antioxidants, vitamins, and minerals that support overall health, boosting hydration, energy, and recovery without relying on synthetic additives or preservatives.
- 2. Sustainability:** Plant-based ingredients typically have a lower environmental footprint compared to animal-derived or synthetic substances. They contribute to a more sustainable production process, aligning with growing consumer demand for eco-friendly products.
- 3. Allergen-Friendly:** Many plant-based ingredients are naturally gluten-free, dairy-free, and vegan, making them suitable for a wide range of dietary preferences and sensitivities.
- 4. Enhanced Nutrition:** Plant-based ingredients such as herbal extracts, fruits, and vegetables can offer functional benefits like improved digestion, immune support, and anti-inflammatory properties.
- 5. Clean Label Appeal:** Consumers increasingly prefer products with simple, recognizable ingredients. Using plant-based components enhances transparency and the "clean label" appeal, fostering consumer trust.
- 6. Innovation and Variety:** Plant-based ingredients allow for unique flavors and formulations, creating new and exciting products that can appeal to health-conscious consumers looking for functional beverages.

What is composition of SmartWater?

1. Packaging

SmartWater is known for its sleek, minimalistic packaging, which is part of its premium positioning. Here's how you could break down packaging:

- **Material:** Primarily PET plastic bottles (often recyclable). SmartWater uses a distinctive clear bottle with a modern, angular shape. This enhances the perception of purity and sophistication.
- **Bottle Sizes:** Common sizes might include 330ml, 500ml, 700ml, and 1L bottles. Each size would have a proportionally sized label.
- **Cap:** Typically a screw-on cap with tamper-evident features. Some versions might have sport caps for convenience in active settings (e.g., gym use).
- **Eco-friendly Considerations:** The packaging should ideally be designed with sustainability in mind, such as using 100% recycled plastic (rPET) or ensuring that it can be easily recycled.

2. Label Design

SmartWater has a clean, modern design. The label usually follows a minimalist aesthetic, reflecting its premium brand image. Key features would include:

- **Brand Logo:** Prominently displayed, likely in a clear, bold font (e.g., “SmartWater” in capital letters).
- **Water Type:** Describing it as “vapor-distilled water with added electrolytes for taste” is critical for distinguishing it from regular bottled water.
- **Nutritional Information:** This includes the ingredients (usually electrolytes like potassium, magnesium, and calcium), along with details on the water’s mineral content.
- **Sustainability Messaging:** Any eco-friendly initiatives, such as the use of recycled materials or the brand’s sustainability goals, would be highlighted.
- **Compliance and Certifications:** Relevant certifications or regulatory symbols such as FDA approval or recycling logos, depending on regional requirements.
- **Design Style:** Typically using transparent labels with a cool color palette (blue or silver tones) to convey a clean and refreshing image.

3. Drink Composition

SmartWater is often marketed as “vapor-distilled water,” meaning the water goes through a distillation process where it is evaporated and condensed, leaving behind impurities.

- **Base Ingredient:** The primary ingredient is purified, vapor-distilled water.
- **Electrolytes:** Added electrolytes include calcium chloride, magnesium chloride, and potassium bicarbonate, which give SmartWater its signature taste.
 - Calcium helps maintain bone health and contributes to hydration.
 - Magnesium aids in muscle function and supports hydration.
 - Potassium helps regulate fluid balance and maintain proper hydration.
- **Taste:** The electrolytes help to provide a slightly "enhanced" taste compared to regular distilled or spring water, giving it a crisp, smooth finish.

Example PLM Process:

- Concept Phase: Research and ideation for potential new bottle designs, improvements in the water distillation process, or introducing new flavor variants.
- Development Phase: Testing water compositions, sourcing sustainable packaging materials, and finalizing label design.
- Production Phase: Once the design is approved, the packaging is finalized, labels are printed, and water is bottled with the correct electrolytes.
- Market Launch: Distribution to various retail and online channels, alongside promotional campaigns.

Ingredient BOM

Structure Manager - Teamcenter 14

File Edit View Tools Window Help

infodba (infodba) - dba / DBA - [IMC--1139590351] [] [] [] []

TEAMCENTER SIEMENS

Structure Manager

Search

Enter the Item ID to search

Quick Links

Home

My Worklist

My Saved Searches

My Links

Open Items

History

Favorites

Getting Started

My Teamcenter

Lifecycle Viewer

Structure Manager

ADA License

CAE Manager

Multi-Structure Manager

Part Planner

Manufacturing Process Planner

Structure Manager

000117/A;1-Ingredients - Latest Working - Date - "Now"

BOM Line	Item Type	Rule configured by	Item Rev Status	Find No.	Mak...	U...	Reference ...	All
000117/A;1-Ingredients	Ingredients							
000118/A;1-Main Ingredients	Main Ingrid	Working()		10			e...	
000123/A;2-Water x 450	Main Ingrid	Working()		10			e...	
000124/A;1-Sodium Chloride	Main Ingrid	Working()		20			e...	
000125/A;1-Potassium Citrate	Main Ingrid	Working()		30			e...	
000126/A;1-Magnesium Sulfate	Main Ingrid	Working()		40			e...	
000127/A;1-Calcium Lactate	Main Ingrid	Working()		50			e...	
000128/A;1-Glucose	Main Ingrid	Working()		60			e...	
000119/A;1-Acidity Regulators	Acid Regulator	Working()		20			e...	
000129/A;1-Citric Acid	Acid Regulator	Working()		10			e...	
000130/A;1-Sodium Bicarbonate	Acid Regulator	Working()		20			e...	
000120/A;1-Flavoring & Sweetening Agents	Flavn Sweet	Working()		30			e...	
000131/A;1-Natural Fruit Flavors	Flavn Sweet	Working()		10			e...	
000132/A;1-Stevia Extract	Flavn Sweet	Working()		20			e...	
000121/A;1-Preservatives	Preservatives	Working()		40			e...	
000133/A;1-Potassium Sorbate	Preservatives	Working()		10			e...	
000134/A;1-Sodium Benzoate	Preservatives	Working()		20			e...	
000122/A;1-Optional Boosters	Opt Boosters	Working()		50			e...	
000135/A;1-Caffeine	Opt Boosters	Working()		10			e...	
000136/A;1-L-Carnitine	Opt Boosters	Working()		20			e...	
000137/A;1-Taurine	Opt Boosters	Working()		30			e...	
000138/A;1-Vitamin B3 (Niacin)	Opt Boosters	Working()		40			e...	
000139/A;1-Vitamin B6	Opt Boosters	Working()		50			e...	
000140/A;1-Vitamin B12	Opt Boosters	Working()		60			e...	

This Bill of Materials (BOM) outlines the composition of a 500mL electrolyte energy drink, detailing ingredients, their functions, suppliers, and nutritional contributions for manufacturing and regulatory compliance. The base is purified water (425-450mL), ensuring hydration and dilution of active ingredients. Sodium chloride (150mg) replenishes sodium, essential for fluid retention and electrolyte balance, while potassium citrate (100mg) provides potassium to prevent muscle cramps. Magnesium sulfate (50mg) supports muscle recovery, and calcium lactate (25mg) aids in hydration and bone health. Glucose (25-35g) serves as a quick energy source, providing 50-60 kcal. Acidity regulators like citric acid (0.5-1g) and sodium bicarbonate (0.25g) maintain pH stability, while natural fruit flavors (0.1-0.25g) and stevia (25mg, optional) enhance taste. Preservatives like potassium sorbate (0.1g) and sodium benzoate (0.1g) ensure shelf stability. Optional boosters include caffeine (15-30mg) for alertness, L-carnitine (0.25-0.5g) for fat metabolism, and taurine (0.25-0.5g) for hydration and endurance. Vitamins B3 (2.5mg), B6 (0.25mg), and B12 (0.5mcg) support energy metabolism and muscle recovery. Overall, this drink provides 12-15g carbohydrates, 150-300mg sodium, 100-200mg potassium, 50-100mg magnesium, and optional stimulants for hydration, electrolyte balance, and energy replenishment.

What is Business Modeler IDE (BMIDE)-

BMIDE is a tool for adding custom data model objects on top of the default Teamcenter data model objects. The BMIDE accomplishes this by separating your data model into its own set of XML files that are kept apart from the standard data model, known as the commercial off-the-shelf (COTS) data model.

Custom data model objects are created and stored in a template and are developed from the COTS data model objects provided by Teamcenter. Therefore, custom templates are dependent on COTS templates. As a matter of best practice, changes should be made only to the custom objects you develop. You may delete from the data model only your custom objects.

A template is a set of XML files that contain the data model for an application, also known as a solution. BMIDE provides a graphical interface to allow you to create modifications to the data model, shielding the user from much of the complexity of writing the XML files. The XML files are rolled into a custom template, and this template is used by Teamcenter to extend and modify the standard data model applied to the database.

One of the most important jobs performed in BMIDE is creating business objects to represent different kinds of parts, documents, change processes, and so on. Business objects are the fundamental objects used to model business data. Companies use business objects to capture the way they conduct their operations and store data about their products.

The screenshot displays the BMIDE application window. On the left, a tree view shows the project structure under 'a7onartwater', including folders like 'Applications', 'Audit Manager', 'Client UI Configuration', and 'Constants'. The main area is titled 'Form : A7Quality_form' and contains a 'Details' tab with various configuration fields. The 'Project' is set to 'a7onartwater', 'Name' is 'A7Quality_form', and 'Display Name' is 'Quality form'. The 'Storage Class' is 'Form', 'Parent' is 'Form', and 'Icon' is 'Default'. The 'Type' is 'Persistent'. There are checkboxes for 'Is Abstract?', 'Is Exportable?', and 'Allow creating instances of Secondary Business Objects'. The 'Teamcenter Component' is 'a7onartwater'. The 'Form Storage Class' is 'A7Quality_formStorage'. The 'Template' is 'a7onartwater'. Below the 'Details' tab, there is a 'Business Object Constants' section with a table of constants.

Name	Value	Overrides	Allow Info...	Allow Over...	COTS	Template	Get...
CreateInput	A7Quality_for...					a7onart...	
DisplayName	Subject_name					foundat...	

Business Object:

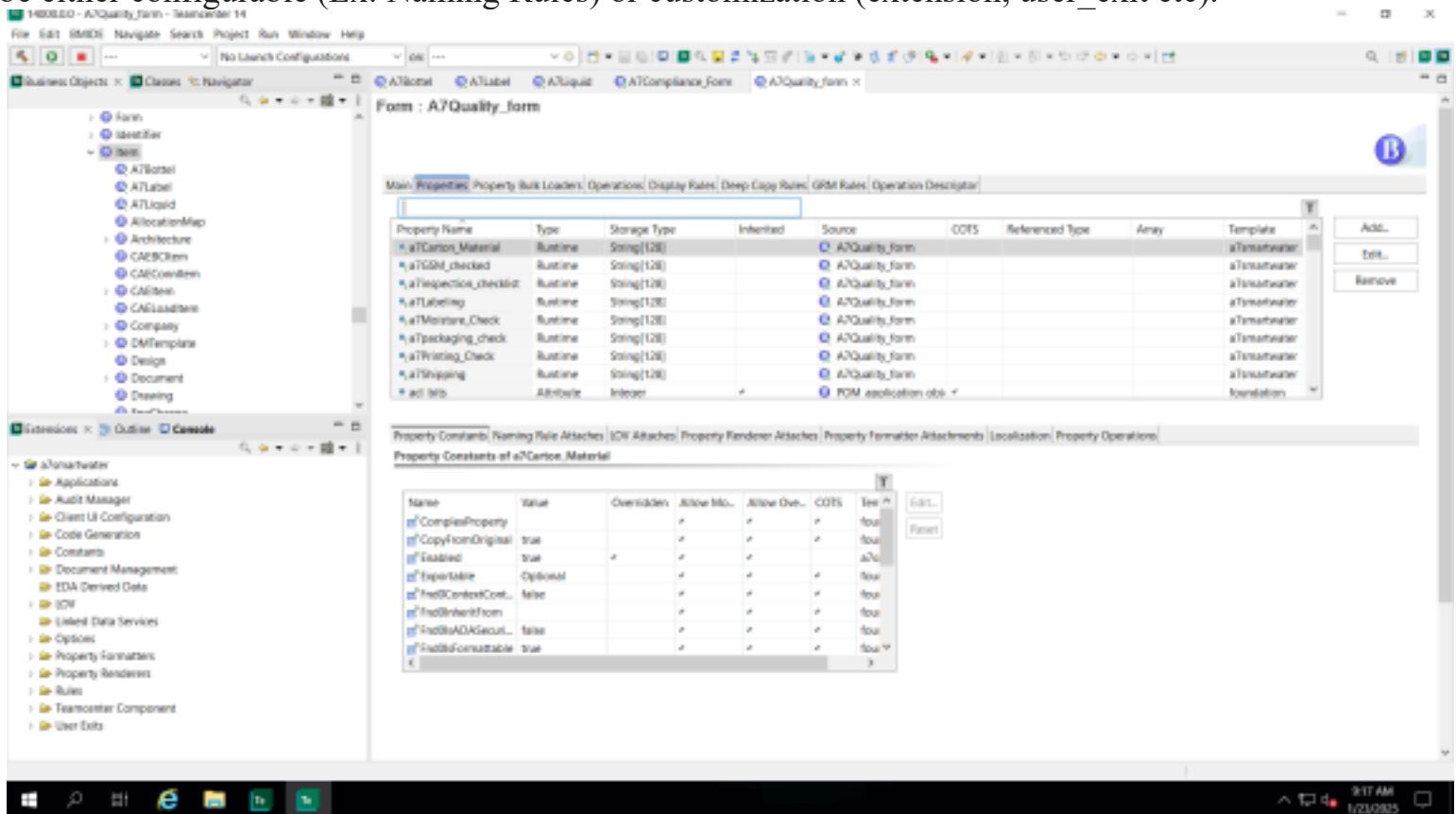
The building block of Teamcenter is Business Object. It resides above POM Objects or DB Classes. Business Object can be seen as actual representation of real life entity which are encapsulated as Business object. The underlining objects are still persistence schema classes. Teamcenter UA

provides hundred of OOTB business objects. Following are major characteristic of Business Object.

1) Business Objects are related to each other through relations.

2) Business Objects have property which can be persistence (attributes from underlining classes) or Dynamic (evaluated run time).

3) Business Objects behavior can be controlled through rules which are defined in BMIDE. Rule can be either configurable (Ex: Naming Rules) or customization (extension, user_exit etc).



Properties

Properties are the attributes that describe the business object.

- Persistent Properties are stored in the database.
- Compound Properties are properties from one object shown on another object.
- Runtime Properties are properties that are computed at runtime.
- Relation Properties are properties that show related objects.
- Table Properties allows you to store structured data in a table format.
- Name-Value Properties display name-value pairs in a tabular format.

Schema

Schema is comprised of Classes and Attributes: the logical database representation of the business objects and properties. Attributes represent your meta-data and store the following data types.

- String

- Integer
- Date
- Float
- Character
- Logical
- Typed Reference
- Untyped Reference

Business Rules

Business rules are the business behavior extension points of Teamcenter.

The following is a list of the behavioral extensions points in Teamcenter. ●

Display Rules

- Naming Rules
- Revision Naming Rules
- Deep Copy Rules
- GRM Rules
- ID Generation Rules

Naming rules shown above define the data entry format for a business object property when the new object is created, for example, when you create a new Revision (Revise) or copy an existing Item. Naming rules can be attached to the properties: Item ID, Item Revision, and Name in Item Types.

Naming rules consist of two components: a pattern and a counter. The pattern is a variable that defines the format (Example: NNNNN), and the counter is used to define the increment each time it is used.

The Item Name and Item Revision Name properties can have more than one pattern defined, the pattern can be conditional, but the pattern is not governed by a generator.

List of Values

List of values provide a pick list of options for a given property in the UI.

Business analyst can set up predefined lists that match the property's data type: String, date, integer, etc. There are three types of LOVs.

- Classic
- Batch
- Dynamic

Classic LOVs shown above store the values in the BMIDE template. They are further classified into two types-

- A Cascading LOV (also known as hierarchical) is a list of values whose values contain other lists of values. Cascading LOV's can be Exhaustive, Suggestive or Range and have multiple entries per level, and be multiple levels deep. The only entry that is written into the property in the database is the end-point (leaf node) value.

- Interdependent LOV's keep track of every entry you pick (a roadmap) and can be Exhaustive or Suggestive. The root note determines the structure's LOV usage type. User defined values are not inserted into the structure.

SmartWater Workflow Process

Group 1: Formula R&D Team

Task: Develop the product formulation and initial BOM ,

Access Permissions: Full Access to Structure Manager for BOM creation and updates.

Access to Workflow Module to initiate approval workflows.

Read Access to QA and Compliance data for alignment.

Process Handoff: After BOM creation, the workflow is sent to QA (Group 3) for testing

Packaging Design by Packaging Team (Group2)

- Task: Develop sustainable packaging solutions.

- Access Permissions:

- Full Access to Active Workspace for packaging design and BOM integration.

Read Access to product formulation BOM for alignment.

○ Write Access to Workflow to update designs and send for QA approval.

- Process Handoff: Updated packaging designs are sent to Marketing Team (Group5).

Testing by QA Team (Group 3)

- Task: Validate product formulation for quality and durability.

- Access Permissions:

- Read Access to BOM and associated test documents.

- Write Access to Workflow for updating test results and

approvals.

- Subscription Management to track changes to the BOM.

- Process Handoff: Feedback is sent to Compliance (Group 4) for regulatory verification.

Verification by Compliance Team (Group4)

- Task: Ensure regulatory compliance and certification.

- Access Permissions:

- Read Access to BOM and QA test reports.

- Write Access to Query Builder and Report Builder for generating compliance documentation.

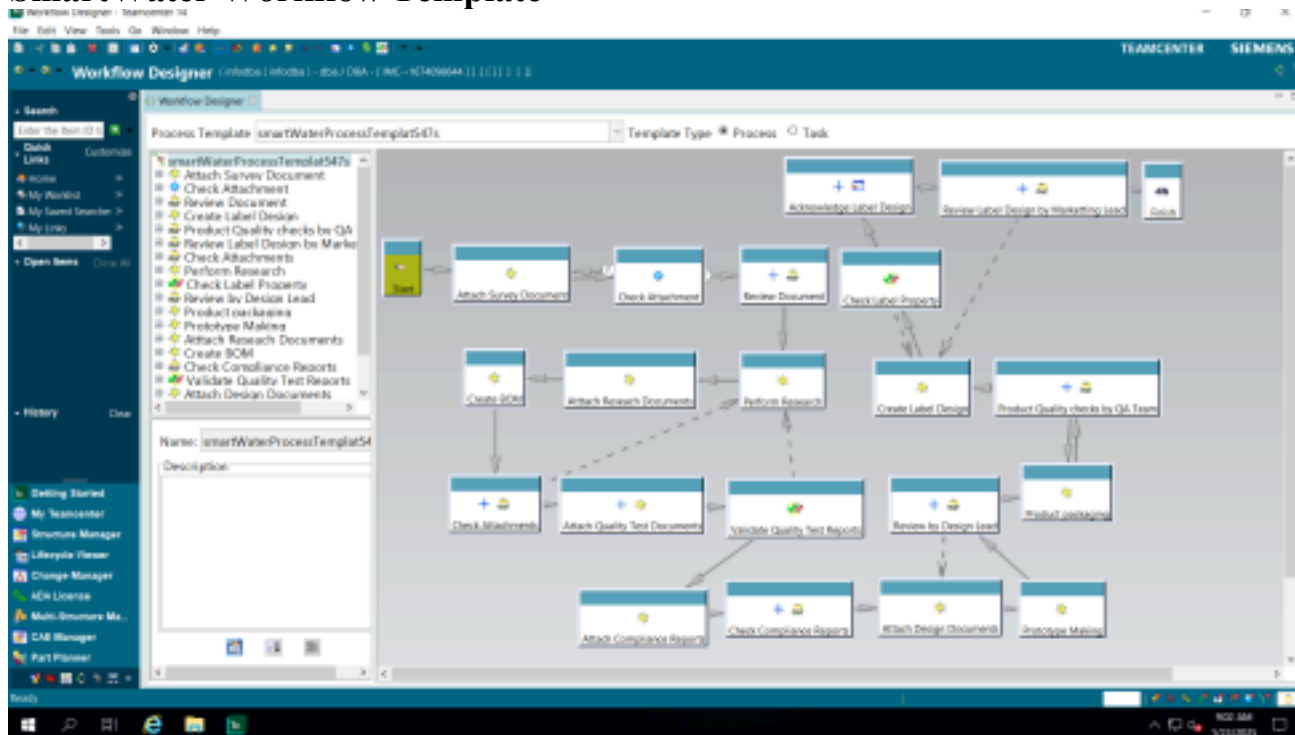
- Workflow Access to approve or reject the product for packaging design.

Process Handoff: Approved workflows are routed to the Packaging Team (Group2).

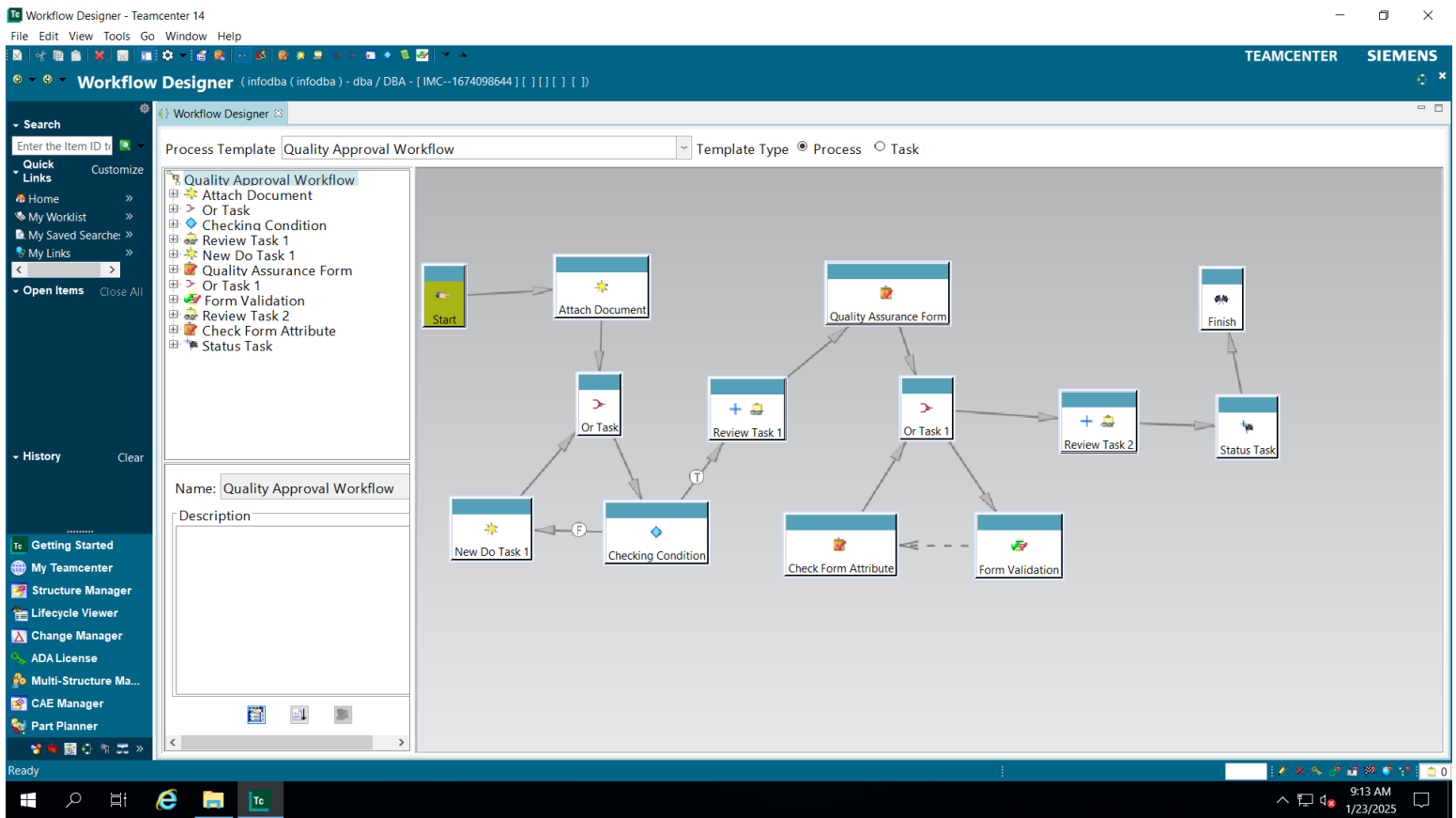
Marketing and Labeling by Marketing Team (Group 5)

- Task: Create product labels and marketing content.
- Access Permissions:
 - Read Access to BOM and packaging designs.
 - Write Access to Workflow for label approval.
 - PLM XML Access to export/import product data for marketing use.
- Process Handoff: Final materials and labels are sent to Change Management (Group6). Iteration and Updates by Change Management Team (Group 6)
- Task: Handle iterative changes based on feedback from all teams.
- Access Permissions:
 - Full Access to Change Management for tracking and implementing updates.
 - Read Access to all prior workflows and team data.
 - Write Access to update BOM, workflows, and subscription alerts.
- Process Handoff: Changes are sent back to relevant groups as needed for further action.

SmartWater Workflow Template

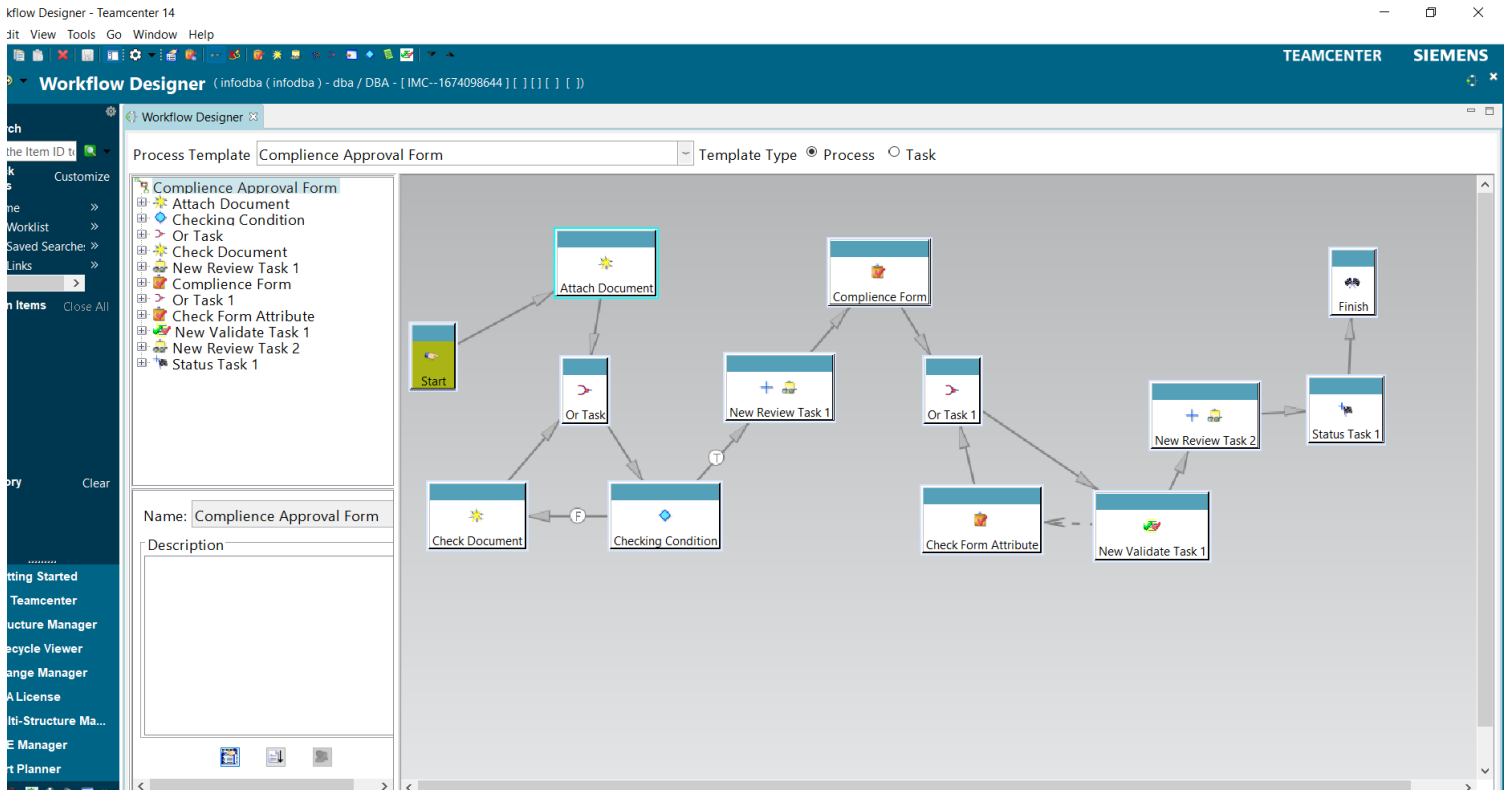


Research and Development Workflow Process-



Compliance Approval Workflow

Process-



Action and Rule Handlers Used

EPM-create-form

DESCRIPTION

Creates an instance of a specified form and attaches that form to the specified task. For more information, refer to the EPM-display-form handler.

SYNTAX

EPM-create-form

ARGUMENTS-

-type

Valid FormType object.

-name

User-defined form name. Default is the workflow process name.

-description

User-defined description of the form. Default value is null.

-property

Specifies the particular field of the form that has a default value. This argument is optional. -value

Specifies the default value for a particular field of the form specified by the -property argument. This argument is optional.

-target_task

Task name and attachment type receiving the new form as an attachment. The default value is the current task.

Accepts one of four keywords for attachment-type:

- \$REFERENCE

Reference attachments

- \$TARGET

Target object attachments

The default value is \$REFERENCE.

EPM-display-form

DESCRIPTION

Displays specified forms attached to a specified custom task, which is an instance of the EPMTaskTemplate. By default, all attachments of the FormType object attached to the current task are displayed.

To create an instance of a specified form and attach the form to the specified task, use the EPM-createform handler.

EPM-display-form handler displays the form when the Perform action is initiated

SYNTAX

EPM-display-form

ARGUMENTS

-type

Valid FormType object.

-source_task

Form to be displayed. The default values for this optional argument are reference attachments of the FormType attached to the current task_name.

-attachment-type

Accepts one of the following reserved keywords:

- \$REFERENCE

Reference attachments

- \$TARGET

Target object attachments

- \$SIGNOFF

Signoff attachments

- \$RELEASE_STATUS

Release status attachments

EPM-hold

DESCRIPTION

Pauses the task, requiring the user to perform an action on the task before the task can complete. Typically, a task completes automatically once started. EPM-hold prevents this automatic completion. To create an instance of a specified form and attach the form to the specified task, use the EPM-createform handler.

Therefore, the EPM-create-form handler creates the form when the Start action is initiated, the EPMdisplay-form handler displays the form when the Perform action is initiated, and the EPM-hold handler prevents the task from automatically completing, allowing the form to be completed by the user.

SYNTAX

EPM-hold

EPM-validate-target-objects

DESCRIPTION

Restricts the types of objects that can be added as target objects. It always prevents the Home, Newstuff, and MailBox folders from being added as target objects.

SYNTAX

EPM-validate-target-objects

ARGUMENTS

-include_type

Defines the type of objects that can be added as target objects to a workflow process. For example, if this argument is specified as ItemRevision, any type of item revision is allowed.

-exclude_type

Defines the type of objects that cannot be added as target objects to a workflow process such as ItemRevision, UGMASTER, and UGPART.

-latest_rev

Ensures any revisions added to the workflow process are the latest revision within their owning item. This argument is optional.

EPM-check-object-properties

DESCRIPTION

Checks that a required or non-null value has been entered for the specified properties of the specified object type that is attached to the current workflow process. If any specified properties do not have the required values, an error message lists those properties. If the specified object type is a form, this handler also checks for form attributes.

SYNTAX

EPM-check-object-properties

ARGUMENTS

-include_type

Specifies the type of the workflow target/reference attachments to be checked. -property

Specifies the properties to be checked. Enter a list separated by commas or the character specified by the EPM_ARG_target_user_group_list_separator preference. -value

Specifies the required real values to be checked. Enter real values as defined in Business Modeler IDE.

-attachment

Specifies the type of attachment to be checked.

- target

Checks the targets attachment.

- reference

Checks the reference attachment.

- schedule_task

Checks the schedule task attachment.

- both

Checks target and reference types of attachments.

If this argument is not used, the target attachment is checked.

-check_first_object_only

If specified, only the first object of type specified by type is considered. This argument is

optional.

EPM-check-condition

DESCRIPTION

By default, this handler is placed on the Complete action of the Condition task, and on the successor tasks of the Validate task. When placed on these tasks, no arguments should be used. When placed on the Complete action of the Condition task, the handler confirms the result of the Condition task is either true or false or the specified custom result. The handler prevents the Condition task from completing until the default setting of unset has been modified to true or false. When placed on the successor tasks of the Validate task, the handler confirms whether errors occurred (either any error, or the specified errors.)

SYNTAX

EPM-check-condition

ARGUMENTS

-source_task

Specifies the name of the preceding Condition task. This argument is required if you place the handler on the Start action of a task succeeding a Condition task. -decision

Specifies the result that must be met. Use this argument in conjunction with a Condition task, placing this handler on a successor task. Valid values are the following:

- custom-result Valid values are any string. When the Condition task's task results return a value matching the value defined for this argument, the successor task starts when the Condition task completes.

- ANY

Use this value in conjunction with a Validate task, placing this handler on a successor task. Indicates that if any error occurs on the Validate task, the workflow process starts the successor task.

- error-code

Use this value in conjunction with a Validate task, placing this handler on a successor task. Indicates that if the specified error codes occur on the Validate task, the workflow process starts the successor task.

EPM-check-related-objects

DESCRIPTION

Checks whether the specified target object contains the required secondary related objects, and whether those objects are in process or have achieved a valid status.

SYNTAX

EPM-check-related-objects

ARGUMENTS

-include_type

Specifies the type of the target object.

-primary_type

Specifies the type of the primary attachment. This argument is mutually exclusive of the -secondary_type argument. You may specify only one of these arguments. -

secondary_type

Specifies the type of the secondary attachment.

-relation

Specifies the relation to be checked.

-allowed_status

Specifies the target object status to be verified:

- Specify any Teamcenter status with ANY.
- Specify no status, or working status, with NONE.
- Specify in process with IN_PROCESS.

-check_first_object_only

If specified, only the first object of type specified by -include_type is considered.

EPM-demote

DESCRIPTION

Clears all signoff decisions from the current and previous Review tasks.

SYNTAX

EPM-demote [-target_task=task-name]

ARGUMENTS

-target_task

Specifies to which previous task the workflow process is demoted. Must specify a valid task in the current workflow process.

If this argument is not specified, the workflow process is demoted to the previous task.

EPM-disallow-adding-targets

DESCRIPTION

Disallows adding targets interactively after a workflow process is initiated. It is good practice to add this handler to the root task Perform action to ensure that target objects are not added from a workflow process once it is started.

SYNTAX

EPM-disallow-adding-targets

What is server-side customization?

Server-side customization in Teamcenter refers to the process of modifying and extending the functionality of the Teamcenter server to meet specific business requirements. Unlike client-side customizations that run on the user's machine, server-side customizations are executed on the Teamcenter server, ensuring centralized processing and better control over the system.

This type of customization is typically implemented using Teamcenter's ITK (Integration Toolkit) APIs, which allow developers to create custom logic that interacts directly with the Teamcenter database and core services. Server-side customizations can automate workflows, enforce business rules, manage data integrity, and integrate with external systems.

Key examples of server-side customizations include defining custom rules (using Rule Handlers), implementing business processes (using Action Handlers), and performing bulk operations (using Batch Utilities). These customizations are designed to run automatically when specific events or actions occur within Teamcenter, such as creating, modifying, or deleting objects.

The primary benefits of server-side customization are scalability, performance, and consistency, as all changes are centrally managed and executed within the Teamcenter environment. This approach also ensures that customizations are applied uniformly across all users, making it easier to maintain and update the system as business needs evolve.

What is ITK in server-side customization?

ITK (Integration Toolkit) server-side customization in Teamcenter refers to the use of ITK APIs to extend and modify the core functionality of the Teamcenter server. Unlike client-side customization, which modifies the user interface or interactions on the client machine, ITK server-side customizations are executed directly on the Teamcenter server, where they can interact with the Teamcenter database and backend processes.

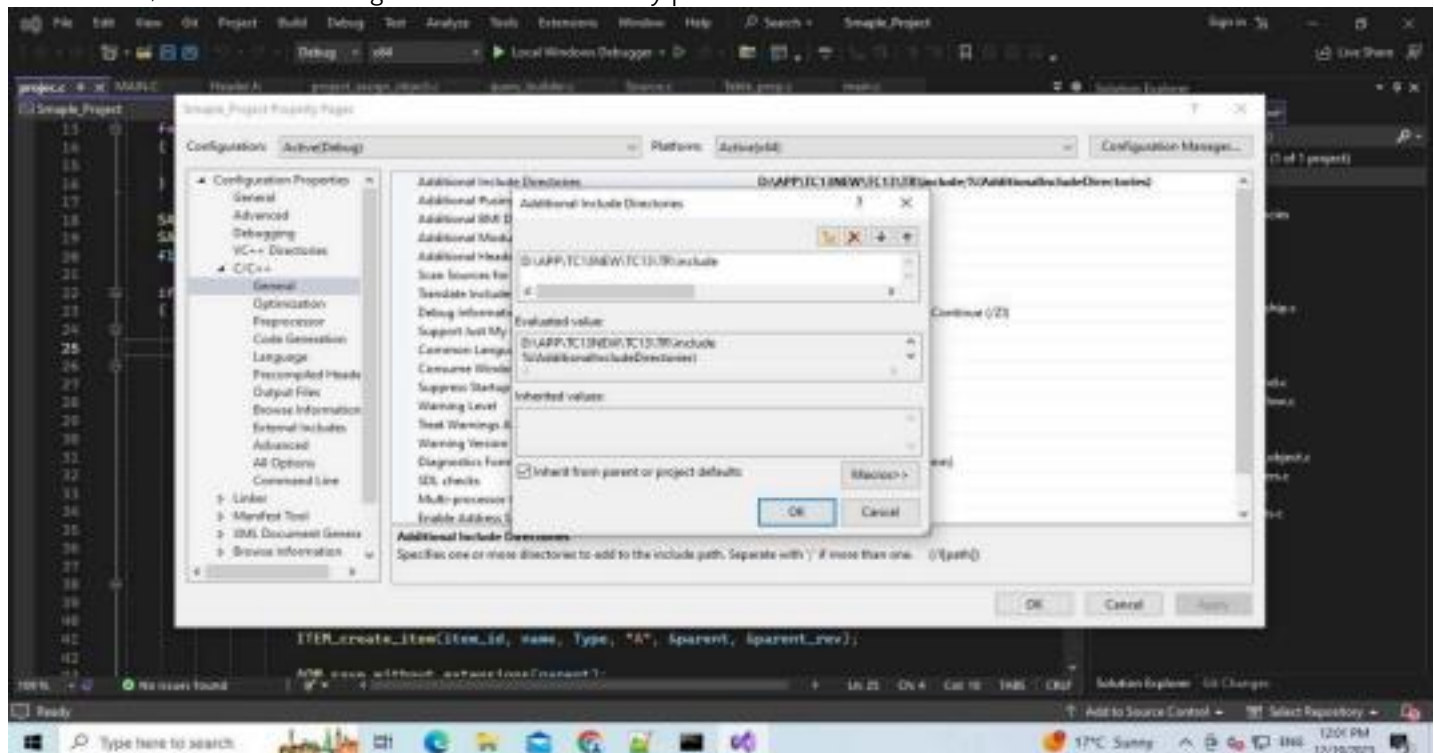
ITK server-side customizations enable developers to automate business workflows, enforce business rules, integrate external systems, and handle complex operations within Teamcenter. Through ITK APIs, developers can define custom logic that runs on the server to manage and manipulate data, trigger actions, or process large datasets.

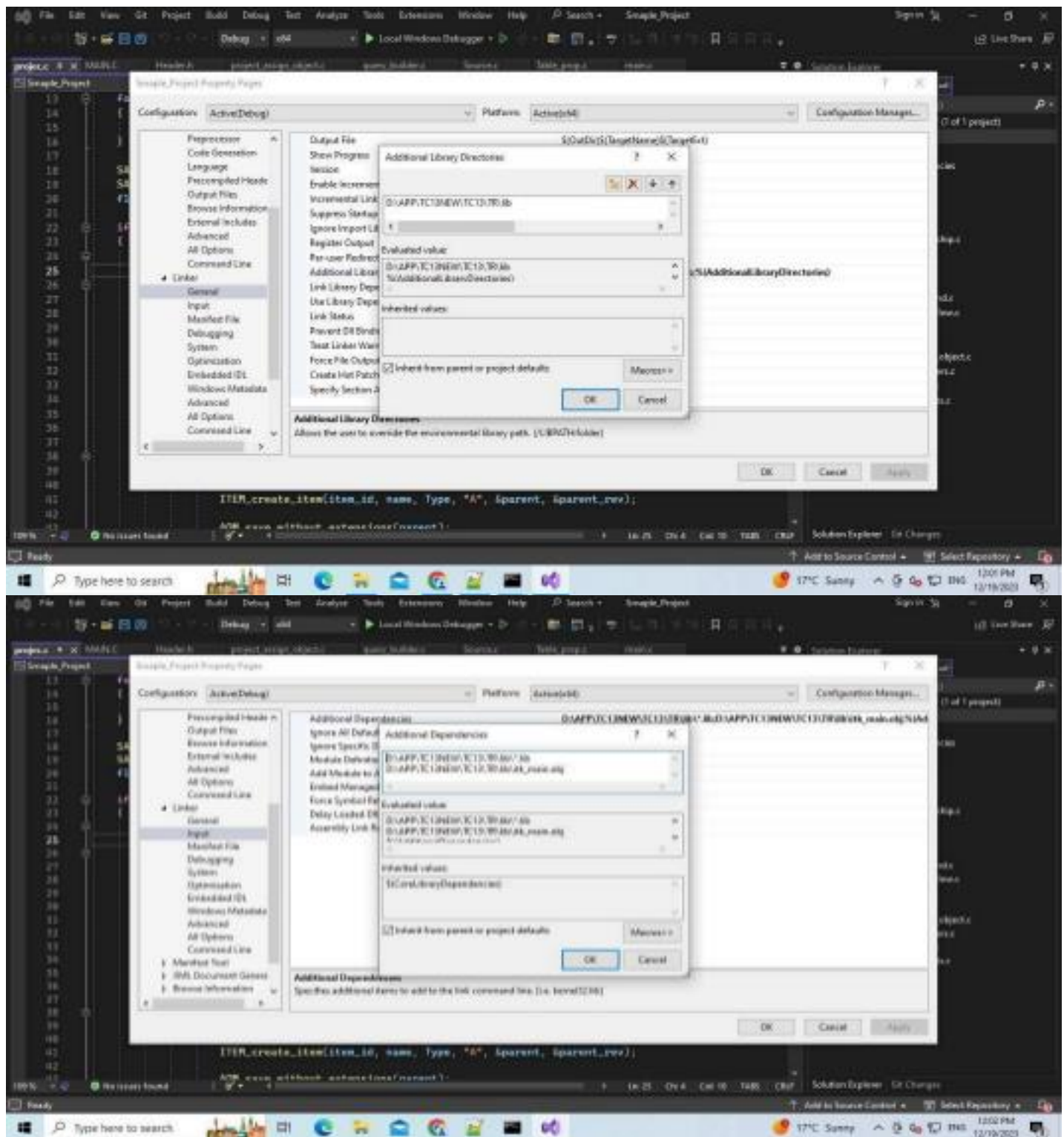
Common examples of ITK server-side customizations include:

- **Batch Utilities:** Automating bulk operations like data imports, exports, or mass updates that are processed on the server.
- **Pre-Conditions and Post-Actions:** Enforcing checks or executing follow-up tasks before or after an operation is performed.
- **Rule Handlers:** Implementing custom validation or business logic that ensures data integrity before certain actions are performed.
- **Action Handlers:** Defining custom actions, such as updating attributes, creating new objects, or notifying users, triggered by specific events in the system.

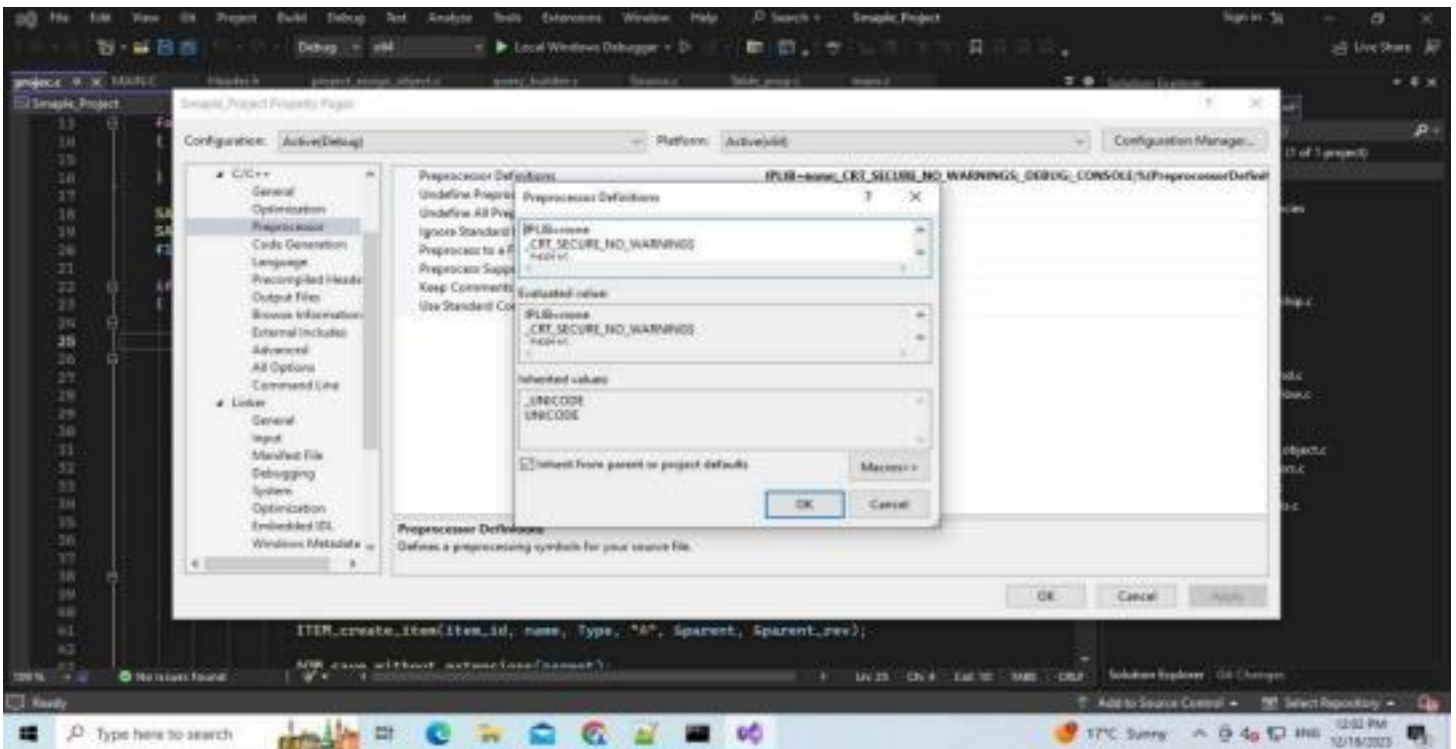
Setting up VS Community for Server Side Customization

1. In Linkers, we have selected general and added library path.



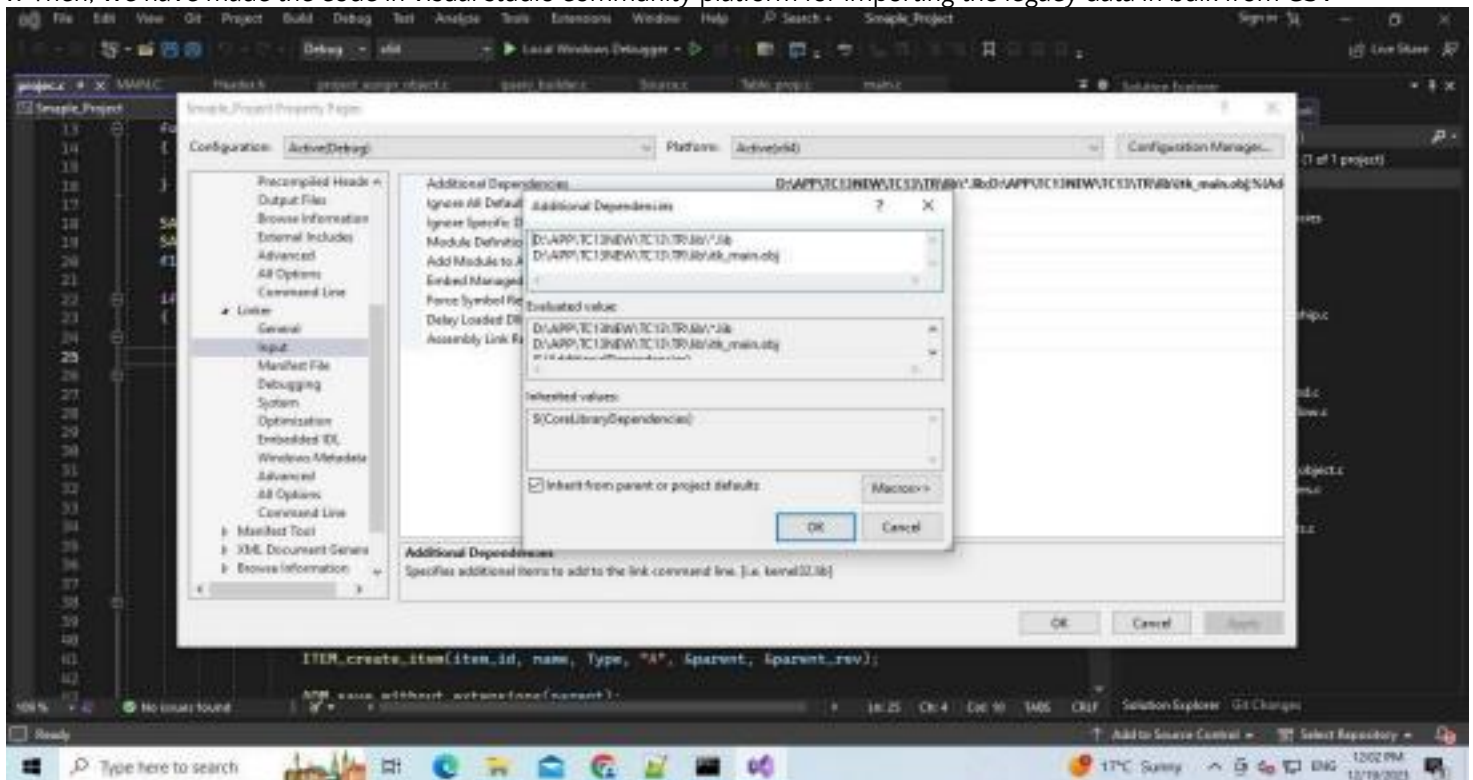


2. In the C/C++, we have selected pre-processor and added IPLIB = none.



3. Added the itk_main.obj in the input which is present in the drop-down of linkers.

4. Then, we have made the code in visual studio community platform for importing the legacy data in bulk from CSV



1} Batch Utility – LOGIN

❖ Result :-

```
tc_config1 Command Prompt
C:\APPS\TC13\TR\tc_menu>cd C:\Users\Admin\Desktop\Somnath_ITK\ITK_programs\Somnath_ITK\x64\Debug
C:\Users\Admin\Desktop\Somnath_ITK\ITK_programs\Somnath_ITK\x64\Debug>Somnath_ITK.exe -u=infodba -p=infodba -g=dba
***Login Successfully***
C:\Users\Admin\Desktop\Somnath_ITK\ITK_programs\Somnath_ITK\x64\Debug>Somnath_ITK.exe -u=infodba -p=infodba -g=dba
The login attempt failed: either the user ID or the password is invalid.
C:\Users\Admin\Desktop\Somnath_ITK\ITK_programs\Somnath_ITK\x64\Debug>Somnath_ITK.exe -u=infodba -p=infodba -g=ba
The given group is unknown.
C:\Users\Admin\Desktop\Somnath_ITK\ITK_programs\Somnath_ITK\x64\Debug>
```

Explanation

The above code is an ITK (Integration Toolkit) utility for logging into a Teamcenter server. The function `ITK_user_main` is the entry point, which takes command-line arguments and performs the login process using Teamcenter's ITK library.

Initially, the function defines several character pointers (`u`, `p`, `g`, and `h`) to store the values of command-line arguments passed by the user. The `ITK_ask_cli_argument` function is used to retrieve these arguments: `-u=` for the User ID, `-p=` for the Password, `-g=` for the Group, and `-h` for Help. If any of these required arguments are missing (i.e., `NULL`), the program prints an error message asking the user to provide the necessary arguments and then returns 0, effectively terminating the utility.

If the necessary arguments are provided, the function proceeds to initialize the Teamcenter module using `ITK_init_module`, passing the User ID, Password, and Group. The return value of this function (`ifail`) indicates the success or failure of the login attempt. If `ITK_init_module` returns `ITK_ok` (indicating a successful login), it prints a success message ("**Login Successfully**"). Otherwise, it uses `EMH_ask_error_text` to retrieve the error message corresponding to the failure code and prints the error message.

Overall, the code provides a basic structure for logging into Teamcenter via ITK, validating input arguments, and handling potential login errors.

Action Handler-

```
#include "Header.h"
int signoffAction(EPM_action_message_t msg) {
    tag_t rootTask;
    tag_t* attachment;
    tag_t* signoffAttachment;
    tag_t sub_task;
    tag_t user_tag;
    tag_t group_tag;
    tag_t member;
    tag_t owningUser;
    tag_t owningGroup;
    int count, numOfArgs, signoffCount, ifail;
    char* argName;
    char* argValue;
    char* obj;
    char* object_type = NULL;
    char* comments;
    EPM_signoff_decision_t decision;
    SIGNOFF_TYPE_t member_type;
    date_t date;
    ifail = EPM_ask_root_task(msg.task, &rootTask);
    ifail = EPM_ask_attachments(rootTask, EPM_target_attachment, &count, &attachment);
    if (count) {
        numOfArgs = TC_number_of_arguments(msg.arguments);
        for (int i = 0; i < numOfArgs; i++) {
            ifail = ITK_ask_argument_named_value(TC_next_argument(msg.arguments), &argName, &argValue);
            if (tc_strcmp(argName, "object_type") == 0) {
                object_type = (char*)MEM_alloc(100);
                tc_strcpy(object_type, "");
                tc_strcpy(object_type, argValue);
            }
            if (argName)MEM_free(argName);
            if (argValue)MEM_free(argValue);
        }
    }
}
```



```

for (int i = 0; i < count; i++) {
    ifail = WSOM_ask_object_type2(attachment[i], &obj);
    if (tc_strcmp(obj, object_type) == 0){
        ifail = EPM_ask_sub_task(msg.task, "select-signoff-team", &sub_task);
        ifail = EPM_ask_attachments(sub_task, EPM_signoff_attachment, &signoffCount, &signoffAttachment);

        for (int j = 0; j < signoffCount; j++) {
            ifail = EPM_ask_signoff_decision(signoffAttachment[j], &decision, &comments, &date);

            if (decision == EPM_approve_decision) {
                ifail = AOM_ask_owner(signoffAttachment[j], &owningUser);
                ifail = AOM_ask_group(signoffAttachment[j], &owningGroup);
                ifail = ITK_set_bypass(TRUE);
                ifail = AOM_set_ownership(attachment[i], owningUser, owningGroup);
                ifail = AOM_save_with_extensions(attachment[i]);
            }
        }

        for (int j = 0; j < signoffCount; j++) {
            ifail = EPM_ask_signoff_member(signoffAttachment[j], &member, &member_type);
            ifail = SA_ask_groupmember_user(member, &user_tag);
            ifail = SA_ask_groupmember_group(member, &group_tag);
            ifail = EPM_ask_signoff_decision(signoffAttachment[j], &decision, &comments, &date);
            if (decision == EPM_approve_decision) {
                ifail = ITK_set_bypass(true);
                ifail = AOM_set_ownership(attachment[i], user_tag, group_tag);
                AOM_save_with_extensions(attachment[i]);
            }
        }
        if (signoffAttachment)MEM_free(signoffAttachment);
    }
}
if (attachment)MEM_free(attachment);
return 0;

```

Description-

The **signofAction** function in Teamcenter is designed to handle workflow sign-offs by verifying workflow attachments, identifying a specific object type, and updating the ownership of attachments based on approval decisions. The function starts by retrieving the root task of the workflow using **ask_EPM_root_task**, then extracts all attachments linked to the task with **EPM_ask_attachments**. It then loops through the workflow arguments to find the "object_type" parameter, which defines the type of object that should be processed.

Next, the function iterates over the attachments to check if their object type matches the specified "object_type". If a match is found, it retrieves the sign-off sub-task, specifically "select-signoff-team", to access the sign-off records. The function then examines each sign-off entry, checking whether the decision was approved (**EPM_approve_decision**). If approval is granted, it extracts the owning user and group from the sign-off entry and transfers ownership of the attachment to the approving user and group using **AOM_set_ownership** and **AOM_save_with_extensions**.

To ensure proper access control, the function temporarily enables bypass mode (**ITK_set_bypass(TRUE)**) before modifying ownership. Additionally, it processes sign-off members, retrieving their user and group details, and reassigns ownership accordingly. After completing these operations, it frees all allocated memory (**MEM_free**) to prevent memory leaks. The function concludes by returning 0, indicating successful execution. This ensures that workflow approvals dynamically update object ownership, streamlining review and authorization processes in Teamcenter PLM (Product Lifecycle Management) workflows.

Conclusion

In conclusion, the development of smart water using plant-based ingredients represents a significant innovation in the beverage industry, aligning with evolving consumer demands for healthier, sustainable, and functional hydration options. By incorporating natural plant-based ingredients, smart water not only offers superior hydration but also provides a host of added health benefits, including vitamins, antioxidants, and electrolytes, all while minimizing environmental impact. This approach appeals to a growing market of health-conscious consumers who prioritize clean, transparent, and eco-friendly products.

Furthermore, the integration of plant-based ingredients into smart water allows brands to differentiate themselves in an increasingly competitive market, offering unique flavors and formulations that cater to various dietary preferences. This innovation also supports sustainability goals, using resources that are renewable and have a lower carbon footprint compared to conventional additives.

As consumer interest in functional beverages continues to rise, smart water with plant-based ingredients is poised to become a leading product in the hydration space. By prioritizing wellness and sustainability, companies can not only meet the demand for more responsible products but also build strong brand loyalty. Ultimately, smart water is more than just a drink—it's a step towards a healthier, more sustainable future in the beverage industry.