Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2024, 11(2):77-84

Research Article ISSN: 2394 - 658X

Product Offering Qualification API Rest Specification

Vijay Kumar Musipatla

ABSTRACT

The TM Forum Open API specification TMF679 standardizes product offering qualification prior to order initiation. This specification facilitates real-time validation. Qualification leverages customer attributes, location data, and business rules. The API employs RESTful endpoints. It supports synchronous and asynchronous operations. Specifically, TMF679 can use notification mechanisms to communicate qualification status changes or provide updates to subscribing systems. The asynchronous capability is crucial for handling long-running qualification processes. Efficient pre-order validation becomes essential. This is due to increasing complexity in telecom operations. Accurate service delivery and enhanced customer experience depend on it. Moreover, seamless integration with BSS/OSS systems is necessary. The integration reinforces operational agility. This paper explores TMF679 core capabilities, structure, and implementation benefits. Dynamic qualification types are examined. The role of TMF679 in digital transformation is also considered. A standardized, scalable approach to TMF679 implementation is proposed. This approach enhances pre-sales efficiency. It ensures accurate serviceability checks. It also supports personalized customer engagement.

Keywords: Product Offering Qualification, TMF679 API, RESTful API Specification, Telecom Service Eligibility, BSS/OSS Integration

---

INTRODUCTION

Delivering frictionless customer experiences requires more than just responsive frontends—it demands intelligent, integrated backend operations. Service providers must enable real-time, data-driven decision-making to ensure that products and services are not only discoverable but also truly deliverable. One of the most persistent operational challenges is the accurate validation of product and service eligibility prior to order placement. Without this validation, service fulfillment failures, provisioning errors, and customer dissatisfaction become inevitable.

To solve this, the TM Forum introduced the Product Offering Qualification API (TMF679)—a RESTful, standards-based interface specifically designed for dynamic, real-time qualification of product offerings. TMF679 empowers Business Support Systems (BSS) and Operational Support Systems (OSS) to evaluate product availability and eligibility using a rules-driven approach across multiple data domains. It eliminates reliance on static logic and manual checks by exposing qualification logic through composable, microservice-aligned endpoints.

Technically, TMF679 defines RESTful operations including POST for initiating qualification requests, GET for retrieving results, and PATCH for updating qualification records. Each request can include key qualification parameters such as geographic coordinates, access technology, customer profile ID, and commercial constraints—ensuring validations reflect actual service feasibility. The API is tightly coupled with service catalogs, CRM systems, network inventory, and policy engines, enabling full-stack orchestration from quote to order.

TMF679 supports both synchronous and asynchronous modes, making it adaptable to various backend latencies and orchestration flows. It plays a pivotal role in scenarios like fiber broadband availability, enterprise VPN serviceability, and location-based mobile promotions. Its flexibility supports multiple qualification types including pre-order feasibility, commercial eligibility, and technical availability checks—all critical in reducing order fallout and improving first-time-right metrics.

Moreover, the API is aligned with the TM Forum Open Digital Architecture (ODA) and the Open API ecosystem, ensuring modularity, vendor-neutral integration, and future-proof scalability. This alignment allows TMF679 to function as a core building block in API-driven ecosystems, supporting service providers in adopting event-based, headless, and AI-assisted operations.

However, successful implementation of TMF679 requires careful planning and alignment with business rules. Qualification criteria must be mapped to backend data sources and updated in real time. Moreover, governance

mechanisms should be in place to track qualification outcomes and refine logic over time. When implemented strategically, TMF679 becomes more than just an API—it becomes a business enabler.
In this paper, we will examine the detailed structure of TMF679, explore its operational lifecycle, and outline implementation strategies, including API notifications, request state transitions, and error handling. We will also discuss its integration with service orchestration layers and the role it plays in enhancing customer experience, improving operational agility, and accelerating time-to-market across the telecom value chain.

LITERATURE REVIEW

The Product Offering Qualification API (POQ) has emerged as a cornerstone for enhancing service validation processes in the telecommunications industry. With the increasing complexity of product bundles and customer-specific configurations, real-time validation has become a critical requirement. The TMF679 specification by TM Forum provides a standard REST-based approach for enabling service providers to determine product availability, eligibility, and feasibility dynamically before order placement [1].
At the core of the TMF679 specification is the principle of RESTful architectural design, which ensures simplicity, scalability, and stateless communication [3]. Fielding's foundational work on REST provides the theoretical underpinning for APIs like TMF679. It emphasizes uniform interfaces, resource identification through URIs, and representations that conform to HTTP standards [3]. Consequently, TMF679 adopts these REST principles to support a wide range of client-server interactions in a consistent manner.
The OpenAPI Specification (OAS) further enhances this approach by enabling automated documentation, validation, and integration testing of RESTful services [4]. By adhering to OpenAPI v3.0.3, service providers can create standardized, machine-readable API definitions. This ensures that TMF679 endpoints are well-described and interoperable across platforms [4].
Moreover, as highlighted in the MEF 87 Developer Guide, the telecommunications ecosystem has expanded its reliance on standardized interfaces to support lifecycle service orchestration (LSO) [2]. MEF’s Cantata and Sonata APIs closely align with TM Forum’s approach, particularly in enabling qualification of services across partners and geographies. This alignment indicates a broader industry trend toward harmonized APIs for service qualification and orchestration [2].
Integration remains a key advantage of TMF679. The API interfaces with critical BSS/OSS systems including inventory, catalog, and CRM platforms [1]. Such integrations allow operators to retrieve real-time customer data, serviceability status, and commercial eligibility—enabling faster decision-making and accurate fulfillment. According to Salesforce documentation, the v5 implementation of TMF679 expands support for complex scenarios like location-based serviceability and discount eligibility [5].
Industry reports like the TM Forum Open API Adoption Assessment Report suggest a growing adoption of standard APIs across service providers worldwide [6]. The report emphasizes that APIs like TMF679 not only improve time-to-market but also simplify partner integration and reduce development costs [6]. Similarly, Capacity Media outlines how these APIs drive automation and innovation in multi-vendor environments [7].
However, implementing REST APIs successfully requires attention to detail. Red Hat notes the importance of proper versioning, authentication, and resource modeling to ensure robust and scalable APIs [8]. Furthermore, Orange’s EVPL Online API documentation illustrates how service qualification APIs are being leveraged in real-world enterprise solutions, providing developers with hands-on examples for implementation [9].
The literature shows strong alignment across industry bodies, vendors, and academic principles in supporting TMF679. From architectural foundations to practical use cases, TMF679 represents a mature, scalable solution for enabling dynamic product offering qualification in digital service ecosystems.

PROBLEM STATEMENT: CHALLENGES IN PRODUCT OFFERING QUALIFICATION API IMPLEMENTATION

Seamless product offering qualification is essential. Enterprises are under pressure to deliver accurate, real-time service availability information across multiple touchpoints. However, implementing a reliable Product Offering Qualification API is far from straightforward. While the TMF679 standard provides a foundational framework, real-world deployments expose several gaps in functionality and integration.
Businesses operating in complex multi-cloud and hybrid environments encounter unique obstacles, especially when aligning API behavior with legacy systems. As customers demand instant, personalized experiences, service providers must overcome technical and operational hurdles to meet expectations without delays. Unfortunately, many existing systems are simply not up to the task.

Lack of Real-Time Validation Systems

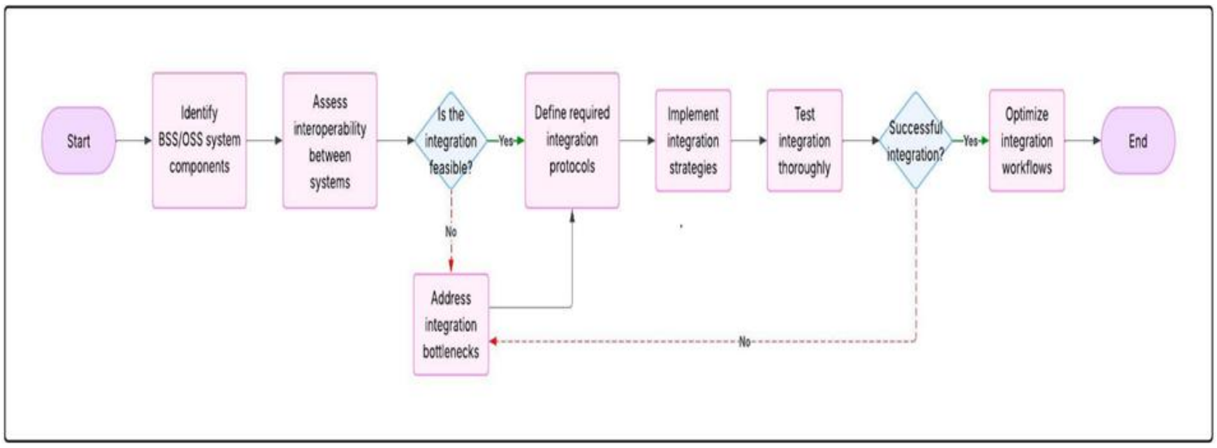
One of the most significant hurdles is the absence of real-time validation mechanisms. Outdated processes rely on static rule sets and manual triggers, which are no longer sufficient. These legacy systems often overlook essential factors such as customer behavior, service history, or regional inventory shifts.

As a result, product availability is frequently misrepresented to end users. This not only disrupts the sales process but also damages customer trust. Consequently, there's an urgent need to transition to dynamic validation models that reflect live data and changing network capabilities. Until then, errors will continue to plague qualification accuracy.

Integration Challenges with BSS/OSS Systems

Interfacing with existing Business and Operational Support Systems presents another layer of difficulty. Despite standardized APIs like TMF679, many systems are not equipped for seamless integration. In practice, workflows tied to catalog management, inventory tracking, and CRM suffer frequent interruptions.

Moreover, each vendor may interpret TMF standards differently, creating misalignments during cross-system communication. This incompatibility increases the complexity of deployment, delays rollouts, and requires extensive custom development. Even minor integration gaps can cause cascading failures across the product lifecycle. Clearly, true interoperability remains elusive in many operational contexts.


Start
Identify BSS/OSS system components
Assess interoperability between systems
Is the integration feasible?
Yes
No
Define required integration protocols
Implement integration strategies
Test integration thoroughly
Successful integration?
Yes
No
Optimize integration workflows
End
Address integration bottlenecks

Limited Support for Complex Qualification Types

Many APIs today are designed with limited scope in mind. While they may support basic availability checks, they often fall short when advanced qualifications are required. For instance, assessing network serviceability or commercial feasibility across product bundles is rarely supported out-of-the-box.

Unfortunately, this limitation forces businesses to layer custom logic on top of standardized APIs. In doing so, they compromise scalability and risk violating architecture principles. A more robust qualification model should support a variety of conditions—technical, geographical, and contractual—without additional patchwork. Without this, achieving accurate and flexible product offerings will remain an uphill battle.

Inconsistent Data Standardization

Lastly, data inconsistency continues to hinder progress. Different systems interpret data fields, formats, and statuses in disjointed ways. This inconsistency leads to duplication, misrouting, and qualification errors during order orchestration.

Moreover, when data flows from multiple sources, mismatches in standards magnify operational inefficiencies. The lack of a unified data governance framework complicates everything from reporting to decision-making. Despite the presence of data mapping layers, errors still leak into customer-facing systems. Therefore, unless organizations align their data definitions and streamline workflows, even the best-designed APIs will underperform.

SOLUTION: ENHANCING TMF679 API FOR OPTIMAL PRODUCT QUALIFICATION

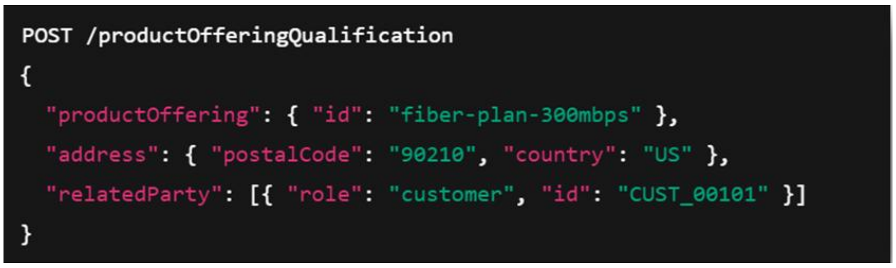
The TMF679 Product Offering Qualification API offers a powerful baseline for service providers aiming to validate whether a product can be offered to a specific customer, at a particular location, and under certain commercial conditions. However, to meet the high standards of today's cloud-native and customer-centric operations, enhancements must be made. These improvements should focus on optimizing real-time responsiveness, interoperability, extensibility, and consistency. A more resilient, flexible, and intelligent implementation of TMF679 is essential to achieving seamless customer experiences and operational efficiency.

To address current limitations, TMF679 should evolve into a more intelligent and scalable qualification engine. This evolution requires enhanced features across data integration, system compatibility, qualification dimensions, and data standardization.

Real-Time Data Integration

Effective product qualification starts with the ability to validate dynamic parameters. Real-time data access ensures accuracy in customer profiling, service location eligibility, and inventory availability. Static lookups and cached datasets introduce delays and errors that ripple through the customer journey.

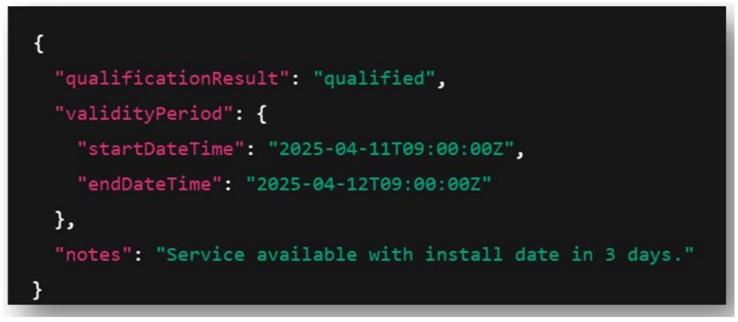
A real-time integration layer can pull live data from various sources including inventory systems, customer relationship management tools, and geolocation databases. For instance, integrating with a network resource function could instantly verify whether fiber service is available at a given postal code.



```
POST /productOfferingQualification
{
  "productOffering": { "id": "fiber-plan-300mbps" },
  "address": { "postalCode": "90210", "country": "US" },
  "relatedParty": [{ "role": "customer", "id": "CUST_00101" }]
}
```

Figure 1: Integrating with a network resource function with JSON

The response from the API should reflect near-instant results pulled from live systems:



```
{
  "qualificationResult": "qualified",
  "validityPeriod": {
    "startDateTime": "2025-04-11T09:00:00Z",
    "endDateTime": "2025-04-12T09:00:00Z"
  },
  "notes": "Service available with install date in 3 days."
}
```

Figure 2: API Response

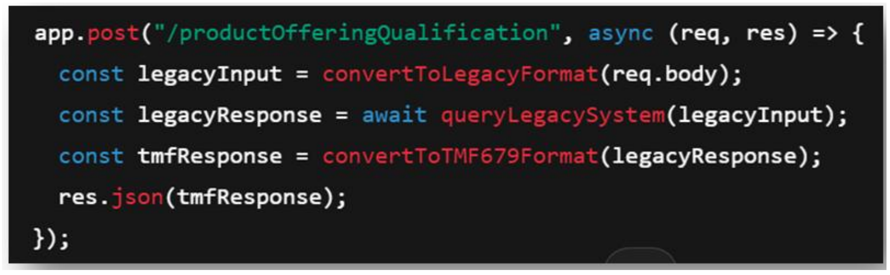
With real-time integration, qualification results are more reliable and timelier. This reduces drop-offs and improves conversion rates.

Seamless Integration with Legacy Systems

Despite digital transformation, many service providers still operate legacy BSS/OSS stacks. These older platforms often lack API readiness or flexible data models. To address this, TMF679 implementations must include middleware solutions capable of translating between modern REST interfaces and traditional protocols like SOAP or JDBC.

A middleware adapter can transform a TMF679 request into a format understood by a legacy system, fetch the necessary data, and translate the result back into a TMF-compliant response.

Consider this logic layer written in Node.js:



```
app.post("/productOfferingQualification", async (req, res) => {
  const legacyInput = convertToLegacyFormat(req.body);
  const legacyResponse = await queryLegacySystem(legacyInput);
  const tmfResponse = convertToTMF679Format(legacyResponse);
  res.json(tmfResponse);
});
```

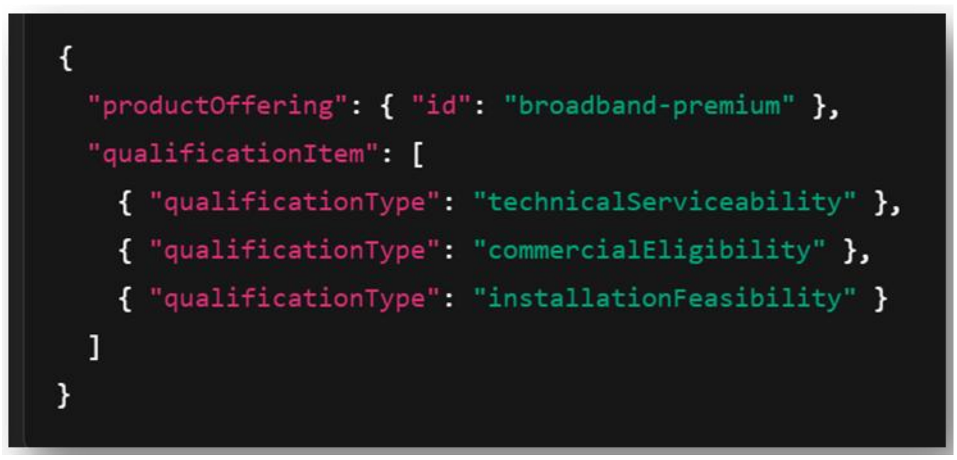
Figure 3: Node.js Javascript code to adopt TMF679

This bridge ensures that organizations can adopt TMF679 gradually without overhauling their infrastructure.

Support for Multi-Dimensional Qualification

Modern qualification demands go beyond simple availability checks. Enterprises must validate technical feasibility, legal restrictions, and commercial eligibility—all within a single API interaction. TMF679 should be extended to support such multi-dimensional assessments natively.

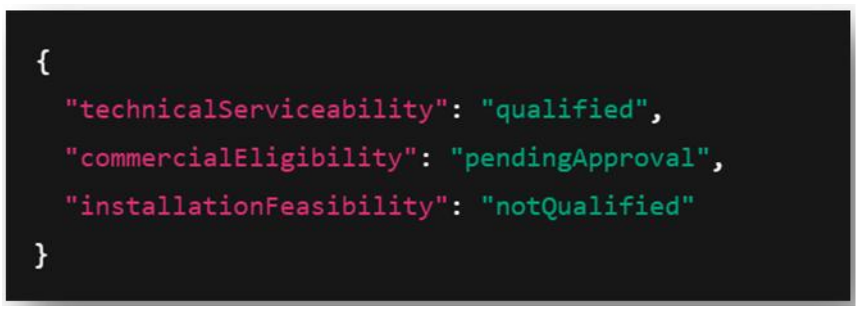
One solution involves enhancing the qualificationItem object in the API schema. This object could include qualificationType, allowing clients to specify the dimension being validated.



```
{
  "productOffering": { "id": "broadband-premium" },
  "qualificationItem": [
    { "qualificationType": "technicalServiceability" },
    { "qualificationType": "commercialEligibility" },
    { "qualificationType": "installationFeasibility" }
  ]
}
```

Figure 4: Enhancing the qualificationItem object in the API schema

Each qualification type can trigger different workflows. Technical serviceability might query a network inventory. Commercial eligibility might validate against credit score rules. A unified response can then return outcomes for all dimensions:



```
{
  "technicalServiceability": "qualified",
  "commercialEligibility": "pendingApproval",
  "installationFeasibility": "notQualified"
}
```

Figure 5: API interaction with Json

Supporting these nuanced checks in one API interaction streamlines the user journey and backend processing.

Establishing Unified Data Standards

Inconsistencies across data formats and systems can derail even the best-designed APIs. Unified data models and standard workflows are essential to ensure accuracy and consistency. TMF679 should rely on tightly defined schemas, canonical IDs, and controlled vocabularies to reduce interpretation errors.

For example, all product IDs should follow the same format across catalog, inventory, and order systems. Validation rules should be centralized to prevent conflicting eligibility decisions.

An ideal setup includes a shared data repository where transformation rules are centrally managed. This allows all systems—regardless of origin—to speak a common language. Additionally, adopting open standards like JSON-LD and OpenAPI for documentation ensures cross-platform transparency and collaboration.

When all parties follow the same data definitions, it eliminates errors during integration, reduces onboarding time for partners, and improves the accuracy of qualification decisions.

Product Offering Qualification API Operations.

Accurate, real-time product availability checks is not just a convenience—it's a necessity. Customers expect seamless interactions across channels, and service providers are under pressure to offer responsive, automated systems that prevent failed orders and enhance satisfaction. A critical part of this process is validating whether a product or service is actually available and eligible before order submission.

To meet this challenge, the **Product Offering Qualification API (TMF679)—**a RESTful interface standardized by the **TM Forum—**plays a central role. TMF679 enables dynamic validation of product offerings by querying real-time backend systems such as network inventory, service catalogs, eligibility engines, and customer profile databases. It allows pre-checks based on geolocation, access technology, customer eligibility rules, and commercial constraints.

TMF679 API Operations

The API defines core operations that support the complete qualification lifecycle:

• POST /productOfferingQualification: Initiates a new qualification request. The request payload includes critical attributes such as product offering ID, geographic address or coordinates, related parties (e.g., customer ID), and any qualification constraints (e.g., bandwidth requirements, SLA expectations).

• GET /productOfferingQualification/{id}: Retrieves the status and result of a qualification request. This includes outcome flags (qualified/not qualified), qualification type, timestamps, and associated validation messages.
• PATCH /productOfferingQualification/{id}: Updates an existing request—for example, to refine input parameters or trigger a requalification.
• GET /productOfferingQualification: Lists or filters qualification requests based on parameters like state, relatedParty.id, or qualificationResult.

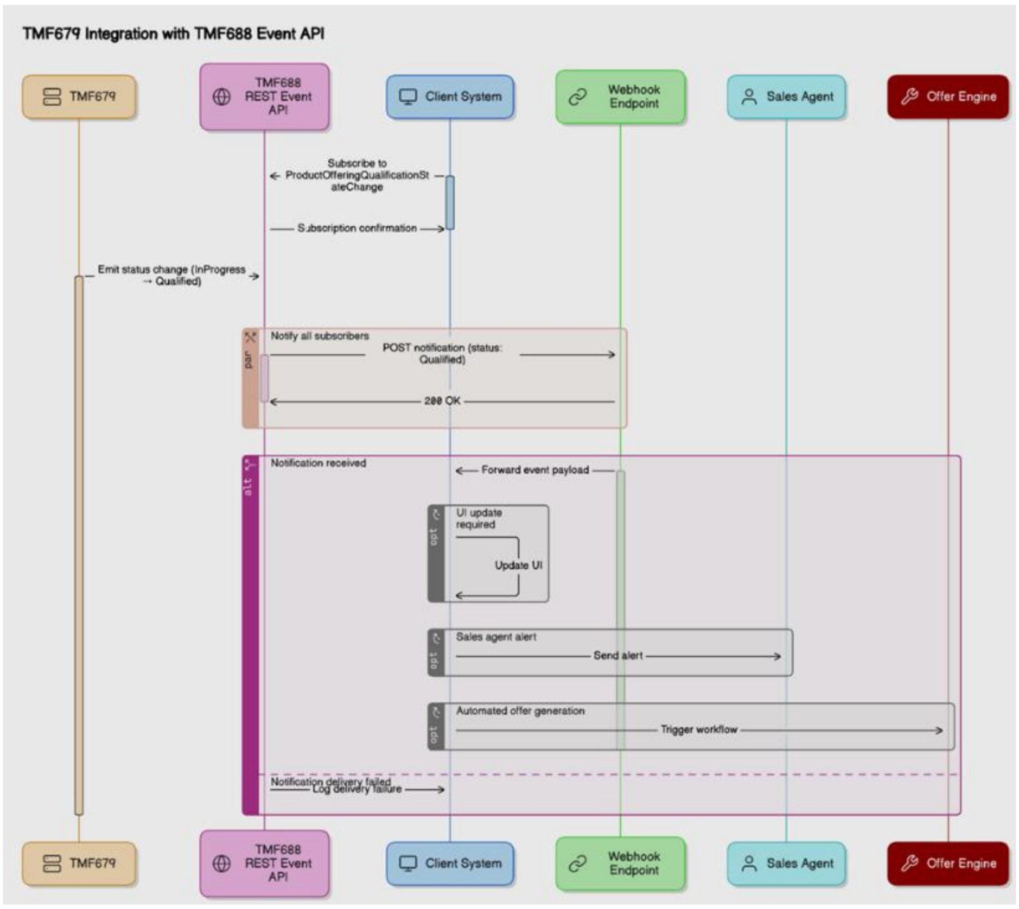
State Transitions & Lifecycle Management

The qualification process supports asynchronous state transitions, including:
• Acknowledged
• InProgress
• Qualified
• NotQualified
• Cancelled
• Failed

These states allow orchestration systems to track each request and trigger subsequent business logic (e.g., notifying CRM agents or auto-populating order forms).

API Notifications and Events

TMF679 integrates with the **TMF REST Event API (TMF688)** to emit status change notifications. This allows client systems to subscribe to ProductOfferingQualificationStateChange events via webhook endpoints. For example, when a qualification moves from InProgress to Qualified, a notification can trigger UI updates, sales agent alerts, or automated offer generation workflows.



Filtering, Search, and Error Handling

The API supports query parameter-based filtering and pagination for managing large qualification datasets. Robust error handling is provided via standardized TMF error payloads, including code, message, and status, which aid in debugging and maintaining system reliability.

Supporting real-time decisioning, extensible qualification logic, and seamless event-based integration allows TMF679 to become more than a functional eligibility checker—it is a foundational technical building block for API-driven service orchestration.

RECOMMENDATION: STRATEGIC STEPS FOR FUTURE-READY LOW-CODE ADOPTION

The low-code development landscape has evolved rapidly in recent years, offering organizations the ability to streamline application development processes. As businesses continue to embrace this shift, low-code platforms are also undergoing significant transformations to meet growing demands for scalability, security, and functionality. The future of low-code development will involve more than just simplifying the development process; it will require careful planning, strategic alignment, and a focus on long-term adoption to realize its full potential.

While low-code platforms offer substantial benefits, their successful implementation and adoption within organizations require careful strategic planning. To ensure low-code tools contribute to the long-term success of a business, organizations must address governance, integration, collaboration, and training challenges. With the right approach, low-code platforms can become an integral part of the organizational infrastructure, enabling faster innovation and increased agility.

Establish Governance Frameworks for Citizen Developers

One of the most significant challenges in low-code adoption is ensuring the quality and security of applications created by citizen developers. These business users, while experts in their respective fields, typically lack formal software development training. Therefore, it's crucial to establish governance frameworks that provide guidelines for app development. Organizations should introduce standards that define best practices for coding, security protocols, and compliance.

Monitoring tools can be used to track the performance and security of apps built by citizen developers. These tools should allow IT teams to oversee development activities and intervene when necessary to address security concerns or performance issues. Moreover, review protocols should be put in place to ensure that all low-code apps go through proper testing and validation processes before they are deployed. A robust governance framework fosters quality control and helps mitigate risks related to data breaches or application failures.

Promote Cross-Functional Collaboration

Successful low-code adoption requires collaboration across various departments within an organization. IT teams, business leaders, and citizen developers must work together to ensure that low-code platforms meet the needs of all stakeholders. To facilitate this collaboration, organizations should encourage alignment between these groups by developing shared roadmaps. These roadmaps should outline the goals, timelines, and expectations for low-code adoption and provide a framework for cross-functional collaboration.

Co-creation efforts should also be promoted, where business teams and IT staff collaborate on the development of key applications. This approach ensures that low-code applications align with business objectives while maintaining the necessary technical standards and security protocols. A culture of cross-functional collaboration not only accelerates low-code adoption but also fosters a sense of shared ownership and responsibility for the success of the platforms.

Scale Training and Cultural Adoption Programs

To ensure successful and widespread adoption of low-code platforms, organizations must invest in comprehensive training programs for their workforce. These programs should go beyond basic tool usage and focus on developing the skills needed to build scalable, secure, and high-performance applications.

Certification programs can be offered to help users gain a deeper understanding of low-code platforms and demonstrate their proficiency. By offering a structured training program, businesses can cultivate a skilled workforce capable of leveraging low-code tools to their fullest potential.

Additionally, cultural adoption programs are essential for fostering a mindset shift within the organization. Employees must be encouraged to embrace the value of low-code platforms and understand how they can empower them to create solutions independently. Community building activities, such as internal forums or user groups, can help spread knowledge and best practices, creating a culture of continuous learning and innovation.

CONCLUSION

Enhancing the TMF679 Product Offering Qualification API requires a strategic focus on agility, extensibility, and consistency. By introducing real-time data integration, ensuring compatibility with legacy systems, supporting complex qualification types, and enforcing unified data standards, service providers can unlock the true potential of product qualification automation. These enhancements not only improve operational efficiency but also elevate the end-user experience—ultimately driving growth and competitiveness in a dynamic digital marketplace.

The future of low-code development is not just about adopting a new set of tools but about rethinking how organizations approach software development. By establishing strong governance frameworks, promoting cross-functional collaboration, and scaling training programs, businesses can unlock the full potential of low-code development. These strategic steps will not only improve the efficiency of app development but also enable organizations to remain agile and competitive in a rapidly changing technological landscape. As low-code platforms continue to evolve, they will play a critical role in shaping the future of software development, making it more accessible and efficient for organizations of all sizes.

REFERENCES

[1]. TM Forum, (2019), "TMF679 Product Offering Qualification API REST Specification R19.0.1", in TM Forum Specification.
[2]. MEF Forum, (2022), "MEF 87 LSO Cantata and LSO Sonata Product Offering Qualification API – Developer Guide", in MEF Standard.
[3]. Fielding, R., (2000), "Architectural Styles and the Design of Network-based Software Architectures", in Doctoral Dissertation, University of California, Irvine.
[4]. OpenAPI Initiative, (2021), "OpenAPI Specification v3.0.3", in OpenAPI Specification.
[5]. Salesforce, (Accessed 2023), "TMF679 v5 Product Offering Qualification Management API", in Salesforce Developer Documentation.
[6]. TM Forum, (2021), "Open API Adoption Assessment Report v3.0a", in TM Forum Exploratory Report.
[7]. Capacity Media, (2019), "October-November 2019 Issue", in Capacity Magazine.
[8]. Red Hat, (2023), "What is a REST API?", in Red Hat Topics.
[9]. Orange, (2023), "EVPL Online API Getting Started", in Orange Developer Portal.