Methodology

The methodologies that will be utilized to manage and deliver the Level Z program are The Delivery Methodology (TDM) and Scrum, which is a lightweight Agile application development process. TDM was built and tested in the field by Client X's expert practitioners and reflects Client X best practices for large-scale Program Management that are also best practices in the Client XYZ coalition. Our proposed methodology and approach for solution development adhere to the guiding principles of Scrum, while also using TDM to mitigate any of the risks that have been associated with some Agile approaches in the past.

At its core, TDM identifies and defines procedures, deliverables, roles, job aids, and checklists for management, strategy, planning, and development of business solutions and program management. The graphic below depicts an overarching view of TDM for Program Management.

[INSERT GRAPHIC]

Our practitioners are skilled at tailoring TDM to meet the needs of specific project activities. Each component and phase of the project activity is broken down into logical sub-components. Each of these components contains the necessary tools, templates, and techniques to drive effective project delivery.

In TDM, the Program Management lifecycle is divided into three primary processes: Plan Program, Mobilize Program, and Manage Program.

- **Plan Program:** Plan Program is an iterative process for developing a high-level program plan. This process is grounded in bottom-up planning from programs with subsequent consolidation to create an integrated plan that captures program interdependencies.

- **Mobilize Program:** Mobilize Program focuses on developing program-level infrastructure. Program management processes are established, the team work environment is developed and deployed, the program management office is established, program-level orientation is developed, and the program resources are identified and deployed.

- **Manage Program:** Manage Program is the ongoing management of the program so that the business capability is successfully delivered and meets documented sponsor and stakeholder expectations. It comprises not only the day-to-day operation of the program, but also the execution of various activities such as schedule, issue, quality, and risk management.

**Plan and Mobilize**

TDM contains an extensive reference library of tools, job aids, document templates, and samples of deliverables. In the initial phases of your Level Z project, we will work to create the right environment for executing and controlling the program.

[INSERT GRAPHIC HERE]
The objective of both the Plan and Mobilize phases is to identify and create the management processes needed to run the program smoothly, develop a program-level plan that will deliver the agreed upon business value, and align the goals and expectations of the program’s stakeholders.

During this time, the Program Management Office (PMO) will be working to:

- Define policies and standards
- Create orientation materials
- Design and implement the physical work environment (layout configuration, desks, computers, phones, etc.)
- Select the vendors and tools
- Establish the PMO and the associated tools and templates
- Obtain and get program resources onboard
- Establish contracts with required vendors

The Client XYZ coalition recognizes that your company has already initiated activity in several of these areas, and we will work with you to mobilize the project as quickly as possible.

Manage Program
The day-to-day management of the Level Z program is the responsibility of the Program Managers and the PMO. Program Delivery is all about creating the right environment for executing and controlling the program. Management of the Level Z program will involve the following PMO functions and activities processes:

- **Status reporting and analysis.** For reporting scope and current progress against the timeline and milestones
- **Issue management.** For issue identification, analysis, escalation, resolution, and reporting
- **Risk management.** For identifying, assessing, and mitigating program risks
- **Financial management.** For establishing budget amounts, tracking expenditures, forecasting, and reporting results
- **Communication management.** For liaison with the Organizational Readiness work stream to plan for communicating messages to program personnel, sponsors, and other stakeholders
- **Resource management.** For sourcing and managing the program’s human and physical resources
- **Configuration management.** For enabling the controlled and repeatable management of information technology components as they evolve in all stages of development and maintenance
- **Release management.** For coordinating and managing the various release components in the program
- **Vendor management.** For selecting and managing vendors and helping to drive consistency of selection and use of vendors across the Level 0 program
- **Quality management.** For selecting the appropriate methods to control the projects that make up the program
- **Knowledge management.** For liaison with Organizational Readiness to ensure that the sharing and leveraging of collective project knowledge occurs across the program
- **Benefits Realization.** For defining, targeting, and tracking key performance indicators against specific benefit categories.
**Agile Scrum Delivery Methods and Development Process With TDM**

Scrum is ideally suited for projects with rapidly changing or highly emergent requirements. Scrum assumes that the software development process is complicated and unpredictable, and treats it as a controlled time box instead of a theoretical, fully defined process like Waterfall and Iterative methodologies. In fact, the Client XYZ coalition has used Scrum on many of our development projects, and that following are the best practices that we developed for implementing Scrum with ADM:

**Best Practice #1: Agile methods are best used in the Build and Test phases**

The use of Agile development does not always lend itself to business user input. During the Analyze and Design phases, it’s important for users to be able to record their requirements to create a baseline. During the Build and Test phases, the development teams possess the ability to make decisions dynamically and validate their decisions against the business requirements and use cases. We implement Agile development by merging detailed design and coding during iterative module development. Business user input is most critical during the Analyze and Design phases and then its criticality lessens with the clarification of requirements and the validation that requirements are being met at the end of each development module.

**Best Practice #2: Architecture design is done up front**

We do not apply Scrum concepts to the implementation of the base application architecture. Our approach is to define the architecture and get it in place, enabling the developers to focus exclusively on the business issues at hand during development, not on worrying about how to build a logging or batch architecture, or about what the Server OS configuration should be.

**Best Practice #3: Development sprints**

Once the product and release backlogs are established, sprint backlogs are peeled off the release stack and a series of sprints approach is executed within a two to four week period. This allows user acceptance of segments of the project’s required functionality, and it remains consistent with a traditional TDM approach, while allowing for more frequent feedback to the users. Our experience has shown that small sprints provide a lower risk and produce high-quality deliverables. With the completion of each sprint, a review is executed and the remaining backlog is used to create a new sprint, as lessons learned are plowed into the process for continual improvement.

**Best Practice #4: Communication and collaboration are key**

Our approach requires a high level of communication and collaboration between business users and sprint teams. Regular feedback between the business units and the sprint teams must include a centralized portal that will house project documentation, requirements, project plans, communication notes, contact information, and the schedule of key module milestones. Frequent, yet brief, team meetings are essential for communicating module status, clarifying requirements, and promoting team unity.

**Best Practice #5: Focus on responding to change, not only following the plan**

TDM provides a change control process that is adhered to within a release cycle. We expect, via the many mechanisms for team communication, to identify changes to requirements, which is another advantage of using short sprints. As changes and issues arise, it is critical to arrive at a quick resolution, normally within 24 to 48 hours, so as not to impact the delivery of the sprint. Also, during the
development sprint, changes can be made instantaneously by the development teams as long as they meet the time box and are germane to completion of the sprint or the release.

Implementing Agile in a distributed organization performing centralized development

Our approach to Agile Development or Scrum makes it possible to have remote business users and a centralized development team. Adherences to the best practices for implementing Scrum within TDM, as listed above, are critical in order for a project to be successful when the users are remote from the development team. Additionally, the following best practices must be implemented for multisite development projects:

1. During the Analyze and Design phases, the development team leads must be involved with the definition of requirements with the business users. It is also important for the development team leads to learn the business processes that are being automated or enhanced to both ensure proper application of the requirements within the business process and to provide input on possible process improvement by using technology.

2. During the Analyze and Design phase, the development team leads must be involved in the development of user cases to ensure proper application of the requirements within the business process.

3. Once the project requirements are base-lined, the development leads and the business users jointly prioritize the backlog, outline interdependencies, and define the backlog sets completed within the release.

4. A member of the development team must remain embedded with the business unit to coordinate and facilitate communication between the developers and business users. The embedded team member is also responsible for evangelizing the project within the business unit, facilitating knowledge transfer, and obtaining user acceptance at the end of each development module.

5. Frequent telephone, video, and/or web conference calls are essential for maintaining open lines of communication. All embedded development team members must ensure that the right users are made available to attend these conference calls, considering that participants may be separated by numerous time zones.

When the best practices that we have outlined are followed, implementing multi-site Agile development projects can prove successful; but they require an investment in embedding the right people with the user community, a flexibility to meeting at inconvenient hours of the day, and a commitment to ensuring that development modules are small in scope and short in duration, so that everyone involved can be kept actively engaged.

TDM and Scrum: A History of Project Success

Together, TDM for Program and Project Management, along with Scrum for rapid solution development, offers the right combination of tools and best practices to drive a successful program. To summarize, our approach provides:
- Proven processes, deliverables, and techniques that enable global teams to define what to do and how to do it
- A comprehensive library of methods that supports multiple types of work
- A focus on the fundamentals and discipline, especially around program and project management
- A common framework to promote growth of consistent skills
- A focus on interaction with end users to deliver rapid turnaround and results
- Repeatable, step-by-step procedures to drive consistency
- Standard language, roles, and deliverables
- A lightweight Agile development process that facilitates frequent and consistent feedback and allows progress to be monitored
Level Z Architecture and Integration Approach
The Client XYZ coalition brings together deep experience, capability, technology, and delivery assets to ensure the successful implementation of the Level Z architecture for your company. We have carefully reviewed the provided materials and understand that the objective of the Level Z program is to develop an architecture on which applications can be built, enhanced, and integrated to meet the operational needs of each SBU and to move technology from a “cost” to an “investment” for the company as a whole.

Our coalition team will adhere to a guiding principle of using your company’s standard architectures, processes, and prefabricated components as appropriate to leverage current investments. In this section of our response, we provide a general overview of how we will implement the Level Z architecture to meet the needs of the SBUs, maximize your company’s investment in technology and move IT from an “infrastructure utility” to a “transformation enabler.”

This is a basic overview of how we will approach the establishment of the Level Z architecture, based on the information we have gleaned from your RFP. Once we are working on the Level Z program, we will conduct an in-depth assessment to determine the specific requirements for each SBU. This will allow us to develop a much more detailed Level Z solution architecture for the SBUs.

Our core responsibility will be to act as an advisor and integrator as we consolidate information systems and data within the SBUs. To integrate these systems, we will use your company’s standard frameworks, packaged applications, and Agile delivery methods. As we have stated, we will adhere to the guiding principle of maximizing current investments by integrating and leveraging preexisting standards and architectural components whenever possible. We will customize, create, and implement new technology only when appropriate to achieve Level Z objectives.

Overview of Level Z Architecture
The Level Z architecture is a blueprint of a service-enabled, loosely coupled software architecture that provides a base infrastructure on which to build the higher level solutions of Level Z through Level ZZ. This approach and architecture address the goals of:

- Using an Agile development approach to adapt to changing business needs of the SBU and OPCO
- Delivering a scalable, available, manageable, and reusable architecture
- Implementing a repeatable development process that ensures the quality of software delivery that is on schedule and within budget.

Our approach to delivering Level Z is to utilize your company’s current process and technology standards, including the Project ABC principles. Thus, the Level Z architecture will be based on service-oriented models. We will take an integration approach to the program by using prefabricated components whenever possible that will include existing company, third party, or Client XYZ coalition assets. And when appropriate, we will create Net-new technology by utilizing our deep expertise to properly implement the Level Z infrastructure.
Overall Needs That Level Z Must Address

Your company’s global upstream objective is to grow profitably in core areas and to build new legacy positions. Supporting this objective requires information that must be available to the right person, at the right time, in the right form, and that is correct and reliable. We will address the following challenges:

- Disparate systems (some performing the same function) that lead to large numbers of interfaces, often highly complex in nature, and considerable maintenance overhead
- Data “silos” that lead to the duplication of data (no single authoritative source), make it difficult to aggregate information to achieve the desired levels of detail, and complicates cross-asset and vendor comparisons
- Deployed, vendor-specific, “single-use” systems that do not offer integration points
- Extensive use of Microsoft Excel spreadsheets that leads to decentralized data and intellectual capital, and, in some cases, difficult compliance issues (e.g., Sarbanes-Oxley)
- Current user visibility tools are “application based” and do not provide ease of user customization

Level Z Development Guidelines

The Client XYZ coalition will adhere to the following guidelines when developing the Level Z architecture:

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Company’s Corporate Principles</td>
<td>Operation Excellence</td>
<td>IT assets must be operationally excellent (safe, reliable, secure, efficient, and effective) and be deployed fully compliant with corporate and governmental regulations</td>
</tr>
<tr>
<td></td>
<td>Standardization</td>
<td>We standardize using common solutions that provide greater long-term value to your company’s bottom line than unique, local solutions</td>
</tr>
<tr>
<td></td>
<td>Integration</td>
<td>We integrate and consolidate information (data, reporting, etc.) across the value chain to enable better business decisions</td>
</tr>
<tr>
<td></td>
<td>Business Process Optimization</td>
<td>We maximize business efficiency, profitability, and reliability by proactively pursuing business process optimization</td>
</tr>
<tr>
<td>Project ABC Principles</td>
<td>Open Platform</td>
<td>The solution architecture should expose services and data in a manner that allows easy access from other platforms and applications without requiring a particular technology. Services and data should be exposed in a manner consistent with open industry standards – such as web services.</td>
</tr>
</tbody>
</table>
### Standards Based

1. **Communications – Protocols for interaction** (TCP/IP, HTTP, XML)
2. **Data Architecture and Integrity:** Standardization of data definitions (i.e., PPDM, WITSML, PRODML) and implementation/integration of trusted data sources
3. **Interfaces – Interaction between services and components** (XML, SOAP, WS specifications)

### Service Oriented and Modular

The architecture will be component-based, with major areas of functionality encapsulated in relatively independent services that will interact through standard interfaces, thus enabling parallel development, modular testing, and replacement of components when necessary.

### Global Scope

Consideration will be given to global consumption so that services will be capable of being expanded worldwide. Due to the complexity of retrofitting such a capability, its support should be considered in the initial design.

The architecture, design, infrastructure, and technologies of the solution should leverage, whenever possible, the existing investment in your company’s governance, data centers, and assets. Deviation in the use of standard guidance and assets must be clearly justified.

**Sources of Policies and Guidance referenced are:**

- Your Procedures Library (GPL)
- Your Security Architecture System (SAS)
- Your Technical Controls (EACT)
- Your IT Architecture

**Lead IT Standards Architect**

We will use the Level Z architecture to build solutions to address the specific requirements of each business unit. Based on your Level Z white paper and RFP materials, we have identified several known requirements. The following section describes how we will use technologies and standards proposed by your company to architect Level Z solutions to meet these requirements. Note that additional requirements and needs may be identified during the course of developing the Level Z solutions. Once identified, we will address these needs through our recommended governance model described in section 2. Furthermore, should we determine that any solution cannot be developed by using the standard company architectures, tools, or processes, the Client XYZ coalition will leverage its capability and assets to build custom components to meet the needs of the your company.