Procurement Portal Application Series

Procurement 3.0 Portal

Developer Setup

Tony Farrell
IT Management Consultant
STATE OF LOUISIANA – DIVISION OF ADMINISTRATION tony.farrell@la.gov

Procurement 3.0 Developer Setup

Summary

This document provides information concerning how a developer could set up the Procurement 3.0 Portal solution in various development environments. **This document is not a user guide.** It is not intended for OTS client or external consumption.

This document contains information about internal architecture and processes. Please safeguard this document accordingly!

Revisions

The current version is 1.0.

Revision	Date
1.0	1/6/2025

Tony Farrell
IT Management Consultant
STATE OF LOUISIANA – DIVISION OF ADMINISTRATION tony.farrell@la.gov

Table of Contents

Tony Farrell
IT Management Consultant
STATE OF LOUISIANA – DIVISION OF ADMINISTRATION tony.farrell@la.gov

Notes on Style (of this document)

- Hyperlinks to the definitions appear throughout the document. The first instance of a definition of each page will link back to the definitions that follow.
- References to figures are linked to the figure and appear in bold type.
- Phrases may be linked to other documents in the Google drive repository.
- Nouns protected by copyright bear the registered trademark symbol, and ownership is acknowledged in a footnote.

Definitions

API	Application Programming Interface – the rules/syntax through which programming actions are to be performed.
CSS	Cascading Style Sheets
EUC	End User Computing, an organization in the Office of Technical Services (OTS)
GIT	Software versioning system integrated with Microsoft® Visual Studio.
GUI or UI	Graphical User Interface: the portion of an application that is exposed to the user; the primary user experience.
IDE	Integrated Development Environment; software like Microsoft® Visual Studio® and SQL Server Management Studio®.
IIS	Microsoft® Internet Information Server – the engine behind Microsoft Web Servers and services.
ISM	XXXXXXXXX; used as a “back end” of P3P.
POC	Proof-of-concept. An application designated as a proof-of-concept is a research project meant to determine if a specific goal may be achieved.
P3P	Procurement 3.0 Portal.
RDBMS	Relational Database Management System; commonly, the “back end” of a Web application.
REST	Representational State Transfer; an architectural style for building web services based on the idea of representing resources using HTTP methods and URIs.
SSMS	Microsoft® SQL Server Management Studio®.

Tony Farrell
IT Management Consultant
STATE OF LOUISIANA – DIVISION OF ADMINISTRATION tony.farrell@la.gov

T-SQL	Structured Query Language (SQL), a programming language used in manipulating relational database management systems. Microsoft's version is Transact-SQL, or T-SQL. Oracle's version is "Procedural Language extensions to SQL," or PL/SQL.

Key Locations

DB SERVER	XXXXXXXXXXX
DEV SHARE	XXXXXXXXXXXXXXXXXXX
GIT	XXXXXXXXXX
ISM DB (DEV)	XXXXXXXXXX
P3P DB (DEV)	XXXXXXXXXXX

Background

The Procurement 3.0 Portal (P3P) began as a proof-of-concept application, with no requirements to build against – just its objective (detailed later). This tasking was received in 2021. The project had no visibility outside of our EUC application development team and certain members of the EUC management hierarchy. The system, still in development due partly to other priorities, represents a significant departure from the standard EUC portfolio.

The idea was to replace the operator-facing forms with a fresh, new UI that takes cues from modern, ubiquitous ecommerce platforms in concept and interface (see **Figure 1**), but maintaining the critical ISM business objects and workflows on the back end.

To date, development of the P3P has been the product of a single developer (the author of this document). Happily, the concept is now proven, and so the project will soon mature away from its POC status.

More information about the background of the P3P may be found in the Technical Overview.[1]

Figure 1. The Landing Page of the P3P application.

XXXXXXXXX

The system was initially called XXXXXXX, which is why so many objects have that in their names. The Web application in IIS is named "XXXXXXX," and all of the back-end objects in the RDBMS have "XXXXXXX" in their names (*spoiler alert*: The folder containing the precompiled code is called "XXXXXXXXXXXXX.")

Development

P3P is architected as a standard Web application with Microsoft® .NET® presentation and business logic connected to a Microsoft SQL Server® RDBMS.[2] [3] The presentation logic includes some client-side code (JavaScript, jQuery libraries) and is styled using CSS. P3P has an application database and connects to the ISM database as needed, primarily through T-SQL stored procedures. Arguably, the heart of the system lay in its ability to communicate with ISM via XXXXXX REST API.

[1] There are actually a few items in this document borrowed from the Technical Overview document.

[2] Microsoft, .NET, SQL Server, SQL Server Management Studio, Office 365, Visual Studio, and ASP.NET are all registered trademarks of Microsoft Corporation.

[3] The application was initially coded in straight HTML with CSS and JavaScript (using Notepad!), with REST capabilities handled in the latter. The decision to port the REST code to .NET was a function of concern for security.

Notes on Coding Format

- Code should be consistent and precise, and well factored.
- Objects should be disposed of as a matter of good practice.
- Each routine should be preceded by an XML summary that describes the routine (lines 381 – 388 in **Figure 2**).[4]
- The Microsoft® standard of using prefixes to identify variable types should be used generally (commonly e.g., “str”, “int”, “dec”, “obj”; lines 394 – 401 in **Figure 2**).
- Comments should also be included to describe processes within routines; they allow developers new to the project the ability to follow along without having to decipher the intent or actions of the process, which gets them “up to speed” more quickly and ultimately saves the organization money.
- Comments should also be included to help distinguish properties from variables (line 416 in **Figure 2**).
- Comments should also appear at the close of every logic block in both T-SQL stored procedures and in .NET code. Notice at lines 425 and 426 in **Figure 2** there are comments following the two End If statements that contain substantial code within the respective logic blocks.

Git

The application code is protected by Git. It is also backed up over the network at regular intervals. The “local” repo I use is located on a network drive.[5] [6]

Database code is also protected by Git, though the process is much more “manual” –

[4] I’m working to include these throughout the application code. I started including them late in the development process.

[5] In most respects, having my “local” repo on a network drive is nice because it benefits from the aforementioned backup scheme and keeps code off of my local machine, but it also means that everything I do requires network resources. The most common byproduct of this choice is lag (I am a remote worker).

[6] If you’re unfamiliar with Git, visit https://git-scm.com.

Periodically, I'll create scripts of the database objects, then place them into a repo via the Web interface.

Setup Requirements

Following is a list of requirements before development on P3P may begin.

1. An account on the XXXXX domain with elevated privileges, which permits access to the DEV share
2. An account capable of accessing the DB server
3. Installation of Visual Studio and SSMS software on your device
4. Installation of Git software on your device.

These are addressed in greater detail later, though I'd like to suggest you get really adept with Git – especially on how to pull, push, and merge.

.NET Development

Because no specifications on system design were provided, I chose to build the system in ASP.NET® because the ISM application it was "shadowing" was hosted in a Microsoft® .NET® environment.

To set up Visual Studio for the P3P solution:

- Download and install Visual Studio 2019 or later if you haven't already. The install may require the assistance of the OTS Help Desk.
- Open Windows Explorer and click on "This PC" in the left hand navigation.

- Create a network location and point it to the DEV share. See **Figure 3**.
- Open the network location using your account with elevated privileges.
- Open Visual Studio.
- On the form that appears following the splash screen, click Open a Project or Solution (Note: with a new install, your form may not look exactly like this). See **Figure 4**.
- You should be able to find the DEV share now in the left-hand navigation, under This PC, XXX (**Figure 5**). Click that.
- Open the XXXXXXXXXXXX folder (also **Figure 5**).
- Click on Project.sln.

You will always need to open the DEV share before attempting to open the project in Visual Studio. Opening the DEV share allows Visual Studio to "see" the drive where the solution is.

To set up Visual Studio for the P3P Git Remote Repo:

- **Follow the procedure outlined in Procurement-3.0-Set-Up-Remote-Git-Repo_202412-30.pdf**.

T-SQL Development

Because ISM uses Microsoft® SQL Server® as its RDBMS, it made perfect sense to create objects to support the new application in the ISM database, then later move those objects out to an independent database with SELECT access to certain ISM database objects.

P3P does not write anything to any ISM DB object. It only ever reads information from ISM.

The database component makes extremely heavy use of stored procedures because they are far more efficient than standard T-SQL queries.

*All communication between the business logic and the database occurs through the **database class**. An instance of the class is created, the method is executed and results from the database are returned as applicable.*

The tool for managing the database development for P3P is SSMS.

Tony Farrell
IT Management Consultant
STATE OF LOUISIANA – DIVISION OF ADMINISTRATION tony.farrell@la.gov

```
''' <summary>
''' Retrieves the data necessary to build OrderData.ItemList,
''' then updates the OrderData record. Make sure OrderData_SummaryCreate() is called first.
''' Requires plaintext version of SessionID as input.
''' Outputs the number of rows updated.
''' Called from Page_Load().
''' </summary>
''' <returns>Integer</returns>
1 reference | 0 changes | 0 authors, 0 changes
Private Function OrderData_ItemListCreate() As Integer

    Dim db As New Database()
    Dim ds As New DataSet()

    Dim intRecordsUpdated As Integer = 0
    Dim strOrderDataID As String = String.Empty
    Dim strAgency As String = String.Empty
    Dim strProcurementType As String = String.Empty
    Dim strOrderedBy As String = String.Empty
    Dim strDateOrdered As String = String.Empty
    Dim strItem As String = String.Empty
    Dim strOutput As String = String.Empty

    ds = db.Database_OrderData_SummaryCreate(SessionID)

    If ds IsNot Nothing Then
        If ds.Tables(0) IsNot Nothing Then

            With ds.Tables(0)
                strOrderDataID = .Rows(0)("OrderDataID").ToString()
            End With

            For Each row As DataRow In ds.Tables(0).Rows
                strItem &= row.Item("Item") & vbCrLf
            Next

            'OrderDataID is a property
            If strOrderDataID <> String.Empty Then
                OrderDataID = strOrderDataID
            End If

            If strOutput <> String.Empty Then
                intRecordsUpdated = db.Database_OrderData_ItemListInsert(OrderDataID, strItem)
            End If

        End If      'ds.Tables(0) IsNot Nothing
    End If          'ds IsNot Nothing

    db.Dispose()

    Return intRecordsUpdated
End Function
```

Figure 2: A typical routine. Notice the XML Summary and the standardized variable naming convention. Also Notice the comments after the long If-End If blocks.

Tony Farrell
IT Management Consultant
STATE OF LOUISIANA – DIVISION OF ADMINISTRATION tony.farrell@la.gov

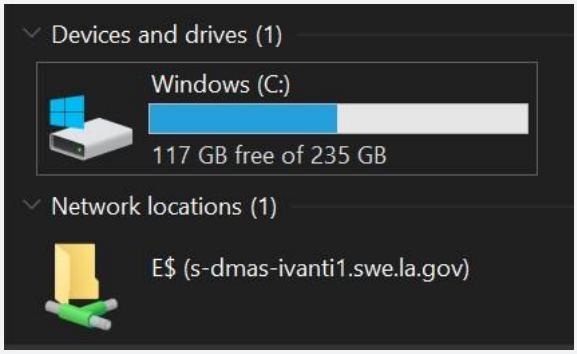

Figure 3. The DEV share as a network location in Windows Explorer.

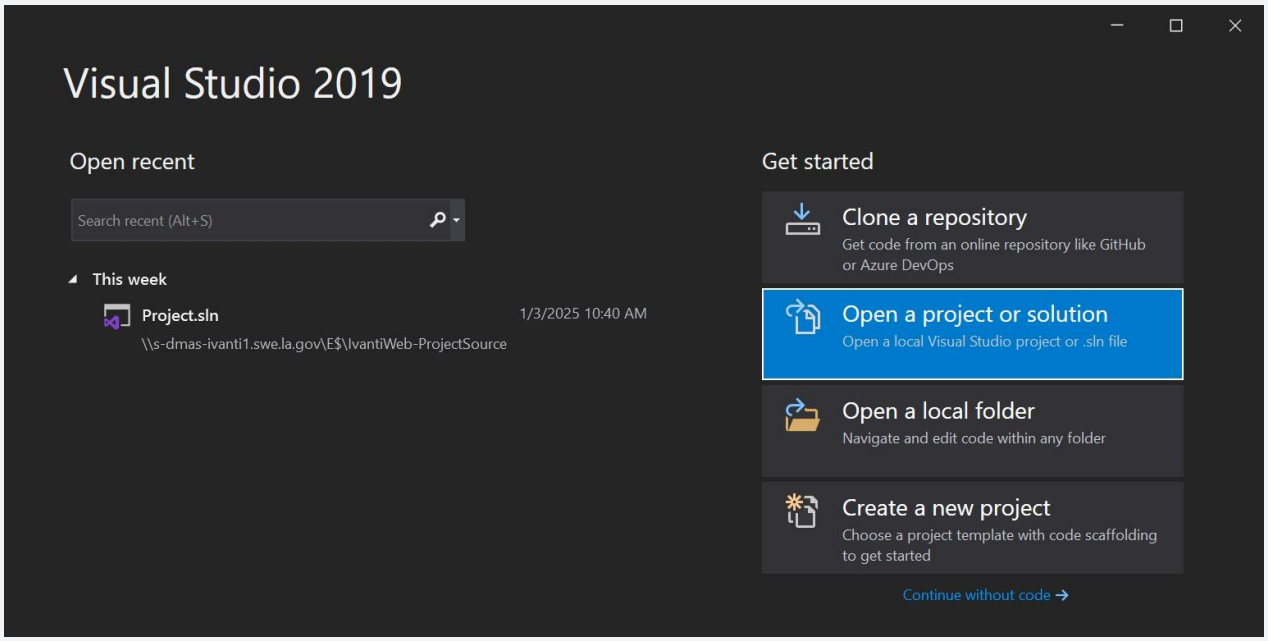

Figure 4. Visual Studio 2019's "Get Started" form.

Figure 5. IvantiWeb-ProjectSource ("graylighted") has the source code for the project. Duh.

To set up SSMS for access to the P3P database:

- Submit a ticket to the OTS Help Desk asking for an account on the DB server, with SELECT rights on XXXXXXX, and full rights on ProcurementPortal. Rights on any environments above DEV (discussed shortly) are not yet required.
- Download and install SSMS version 20 or later if you haven't already. The install may require the assistance of the OTS Help Desk.
- Once you have received credentials, connect to the DB server as follows (these fields are on the Login tab; no other information is required on any other tab; see **Figure 6**):
  - Server Name: XXXXXXXXXX
  - Authentication: SQL Server Authentication
  - Login: {*Your credential*}
  - Password: {*Your credential*}
  - Encryption: Mandatory
  - Trust server certificate: Checked

Figure 6. Connect to the DEV instance of SQL Server

Environments

Technically, there are four environments: The **local** environment is where the application is primarily built and debugged. After testing, code is promoted to the **development** environment. Here the application is launched from the ISM development environment and may be launched by any person with ISM access. The **testing** and **production** environments complete the four, although the application isn't in either of those environments yet.

The application's Web.Config file contains an Environment variable in its AppSettings. This node is chiefly responsible for telling the application what environment the instance is operating in. This is important because access is cued in part from this value. Care must be taken to ensure the Web.Config is not overwritten when code is pushed up from a lower environment.