**TECHNICAL MEMORANDUM**

PRGFX – Multiple Data Technologies Specification

|  |  |
| --- | --- |
| **PREPARED FOR:** | Arnel Mandilag  Arnold Engelmann |
| **PREPARED BY:** | D.F. Cautley |
| **COPIES:** | PRGFX team |
| **DATE:** | June 9, 2010 |
| **Revisions** |  |
| **1.1 -- DFC** | Add placeholder for non SQL data sources and an ASM Emulation |

# Introduction

The PRGFX project is intended to provide a re-usable programming framework for future System Analysis development work. One of the key functions is standardization of the data access methods and the use of a 3rd-party, quasi-commercial framework – NetTiers.

The challenge is that various data subjects will be stored using different data technologies; and that these technologies can migrate over time and in the worst case, from project to project. Thus, the off-the-shelf data access framework, while very rich, will not be sufficient. NetTiers is designed to access to one data storage technology at a time, primarily SQL Server.

# Summary

NetTiers provides a useful, non ADO.NET business object model for a targeted relational database. It also provides a configurable, flexible data access stack between the business object model and the persistence or storage of the data in the RDBMS. We would like to present a single logical business object model to the programmer, who can ignore the technical or physical details around how those objects are persisted. The standard NetTiers provides much of that separation of concerns. We would like to expand that to include the actual physical data storage and technology used to persist that data. In other words, the programmer should not need to know or care that some of the business object model is persisted in SQL Server, some in ESRI SDE, some in one of the file based geodatabase formats, or even MS Access. The ability to read or write to an Excel workbook should be considered, but obviously this use does not hold the same place in the logic priority as the relational data stores.

## Expected Solution

First of all, these comments are NOT intended to constrain the solution.

* We don’t expect to be able to generate code for all the persistence technologies in one pass.
* We do expect that “some assembly is required” to combine artifacts from several generation passes.
* We do expect that a single logical model will need to be constructed, and that the entities (business objects) layer will be generated from that model.

We should not assume that NetTiers is incapable of the functions we need; it may simply be obscure (un-documented).

## Expected Outcome

1. A documented design of the architecture modifications to .NetTiers that enables support of multiple data persistence technologies within the same business object (Entity) model.
2. Modified .NetTiers templates to support the modified architecture.
3. Modified .NetTiers documentation describing the modified architecture and a cookbook for using it.

# General Feature Universe



Figure -- General Feature Universe

# Requirement Statements

**General.**

| Name |  | Notes |
| --- | --- | --- |
|  | **Status:**  **Difficulty:**  **Priority**: | Using the app.config file on every desktop will not be a good solution to storing the default connection strings for various entities.   * Look at whether NetTiers can get config data from a service already * If not, some kind of service for defaults is probably necessary. * Local config would be an override (based on user preferences for a particular configuration or "run" of the application. |
|  | **Status:**  **Difficulty:**  **Priority**: |  |
| Physical Sources Required | **Status:**  **Difficulty:**  **Priority**: | The following databases need to be supported:   * SQL Server * ArcServer SDE (on SQL Server) * MS Access |
| Single Entity Model | **Status:**  **Difficulty:**  **Priority**: |  |

**Spatial Consideration.**

| Name |  | Notes |
| --- | --- | --- |
|  | **Status:**  **Difficulty:**  **Priority**: | Support for spatial operations, even when working with SDE or MapInfo data are not required:  No need to support spatial operations or spatial SQL. |
| Spatial Handline | **Status:**  **Difficulty:**  **Priority**: |  |

# Details

## Structure of a NetTiers generated data access layer (DAL)

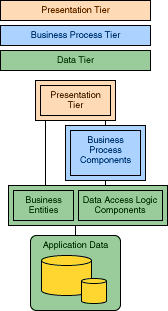
Consider this figure from the .NetTiers documentation. We want all the layers except the very bottom one to be independent of the technology used to store the data.

Figure -- NetTiers Layering

At the Data level, the programmer calls a generated class of the form <Entity>Service; for example

Mst\_LinksService ms = new MstLinksService();

TList<Mst\_Links> mlList = ms.GetAll();

The difference in the physical storage needs to lie at least below the service level. Let’s keep drilling down.

The service calls a data provider this way:

try

{

transactionManager = ConnectionScope.ValidateOrCreateTransaction(noTranByDefault);

dataProvider = ConnectionScope.Current.DataProvider;

list = dataProvider.Mst\_LinksProvider.GetAll(transactionManager);

}

So, the “switch” that selects the physical storage is likely to be buried inside the transaction manager, the data provider, or both.

## Clues

Ideas to follow that could be useful.

1. In the documentation, look at the section captioned Dynamic Connection String
2. In the docs, consult the section Multiple NetTiers Service Sections