25th ESUG Conference
Maribor, Slovenia
September 8, 2017

Enhancing ENVY/Developer for Modern Programming Environments

Seth Berman
Chief Operating Officer
Instantiations, Inc.
Overview

- ENVY Enhancements
- 64-bit ENVY
Enhancements
Making the case

• The Good
  • Stable codebase for many long years
• But...
  • Has not seen many enhancements
• Because...
  • High-Risk Area to change
  • Challenging Codebase Organization
  • Code Duplication
  • Black Box testing only
• Therefore...
  • We were all afraid to touch it
Enhancements
Making the case

• However we would still like to...
  • Have ENVY Pluggable Backends
  • Provide Unicode Support
  • Offer SSL/TLS Encryption for EMSRV
  • Provide “better” WAN Support
  • In General...not fear our codebase
ENVY Pluggable Backends

EmLibrary Backend Interface

- EmLibrary is a record-oriented repository
- EmLibrary store on disk is...
  - A regular file
  - Contains a header
  - Followed by a bunch of ENVY records that are chained together
- An ENVY record...
  - Contains a header
  - Followed by a variable length amount of “stuff”
  - EmClassDefinition, EmMethodEdition...
ENVY Pluggable Backends
EmLibrary Backend Interface

```c
struct _emlibrary_backend {
    U32 type;

    /* Lifecycle */
    I32 (*open)(EmLibraryBackend *self, EMLIBRARY_HANDLE *libraryPtr...);
    I32 (*create)(EmLibraryBackend *self, PCHAR libraryName, U32 accessFlags);
    I32 (*close)(EmLibraryBackend *self, EMLIBRARY_HANDLE library);
    I32 (*free)(EmLibraryBackend *self);

    /* General I/O */
    I32 (*flush)(EmLibraryBackend *self, EMLIBRARY_HANDLE *libraryPtr...);

    /* Stream I/O */
    I32 (*read)(EmLibraryBackend *self, EMLIBRARY_HANDLE library...);
    I32 (*write)(EmLibraryBackend *self, EMLIBRARY_HANDLE library...);

    /* Record I/O */
    I32 (*readRecordSize)(EmLibraryBackend *self, EMLIBRARY_HANDLE library...);
    I32 (*readRecordData)(EmLibraryBackend *self, EMLIBRARY_HANDLE library...);
    I32 (*writeRecords)(EmLibraryBackend *self, EMLIBRARY_HANDLE library...);

    /* Queries */
    I32 (*getMaxSize)(EmLibraryBackend *self, EMLIBRARY_HANDLE library...);
    I32 (*getEof)(EmLibraryBackend *self, EMLIBRARY_HANDLE library...);

    /* Utility */
    I32 (*compareLibraries)(EmLibraryBackend *self, PCHAR libraryName1, PCHAR libraryName2);
};
```
ENYY Pluggable Backends

ENVY Backend

```c
struct _emlibrary_envy_backend {
    EmLibraryBackend parent;
};

EmLibraryBackend *emlibrary_envy_backend_new() {
    EmLibraryBackendEnvy *backend = (EmLibraryBackendEnvy *)calloc(1, sizeof(EmLibraryBackendEnvy));
    if (!backend) {
        return (EmLibraryBackend *)NULL;
    }
    backend->parent.type = EMLIBRARY_BACKEND_ENVY;

    backend->parent.getMaxSize = &envy_backend_maxSize;
    backend->parent.flush = &envy_backend_flush;
    backend->parent.create = &envy_backend_create;
    backend->parent.open = &envy_backend_open;
    backend->parent.close = &envy_backend_close;
    backend->parent.free = &envy_backend_free;
    backend->parent.getEOF = &envy_backend_eof;
    backend->parent.read = &envy_backend_read;
    backend->parent.readRecordData = &envy_backend_readRecordData;
    backend->parent.readRecordSize = &envy_backend_readRecordSize;
    backend->parent.write = &envy_backend_write;
    backend->parent.writeRecords = &envy_backend_writeRecords;
    backend->parent.compareLibraries = &envy_backend_compareLibraries;

    return (EmLibraryBackend *)backend;
}
```
ENVY Pluggable Backends
SQLite Backend

```
struct _emlibrary_sqlite_backend {
    EmLibraryBackend parent;
    sqlite3 *db;
    sqlite3_stmt *st_write_library_header;
    sqlite3_stmt *st_read_library_header;
    sqlite3_stmt *st_read_record_max_id;
    sqlite3_stmt *st_read_eof;
    sqlite3_stmt *st_write_eof;
    sqlite3_stmt *st_write_record;
    sqlite3_stmt *st_read_record_size;
    sqlite3_stmt *st_read_record_data;
};

EmLibraryBackend *emlibrary_sqlite_backend_new() {
    EmLibraryBackendsSQLite *backend = (EmLibraryBackendsSQLite *)malloc(1, sizeof(EmLibraryBackendsSQLite));
    if (!backend) {
        return (EmLibraryBackend *)NULL;
    }
    backend->parent.type = EMLIBRARY_BACKEND_SQLITE;

    backend->parent.maxSize = sqlite_backend_maxSize;
    backend->parent.flush = sqlite_backend_flush;
    backend->parent.create = sqlite_backend_create;
    backend->parent.open = sqlite_backend_open;
    backend->parent.close = sqlite_backend_close;
    backend->parent.free = sqlite_backend_free;
    backend->parent.eof = sqlite_backend_eof;
    backend->parent.read = sqlite_backend_read;
    backend->parent.readRecordData = sqlite_backend_readRecordData;
    backend->parent.readRecordSize = sqlite_backend_readRecordSize;
    backend->parent.write = sqlite_backend_write;
    backend->parent.writeRecords = sqlite_backend_writeRecords;
    backend->parent.compareLibraries = sqlite_backend_compareLibraries;

    return (EmLibraryBackend *)backend;
}
```
ENVY Pluggable Backends

Next Steps

• But there is a problem...
  • The “schema” only exists in Smalltalk

• Backends see records and bytes
  • No knowledge of record interpretation
  • This is historically on purpose...but is no longer a necessary constraint

• No Schema...kinda pointless
  • Storing blob files in GIT would probably not be very useful

• Schema must be accessible at this level
  • Extend EmLibraryBackend interface with record -> schema backend adapter
ENVY Pluggable Backends

GIT Backend

- Hmm...
- Record/Stream Oriented Interface
- Lots of volatile metadata
64-bit ENVY
The Easy Part

- 64-bit EMSRV
  - Recompile with minor changes
- 64-bit ENVY/Client library primitives
  - Recompile with minor changes
64-bit ENVY
The Hard Part

• Serialized Objects in the ENVY/Manager
  • 32-bit format
  • Multiple Versions

• 64-bit Images must be able to...
  • Read 32-bit serialized objects
  • Write 32-bit serialized objects
**64-bit ENVY Translation Engine**

- **32-bit <-> 64-bit Translation Engine**
  - Executable to convert 32-bit images and ICs offline
  - Convert 32-bit images and ICs at runtime
  - Read/Write 32-bit serialized objects
64-bit ENVY Bytecodes

• Bytecodes in the ENVY/Manager
  • Bytecodes are cached in the manager
  • Linked directly into a CompiledMethod
  • Bypasses source compilation
  • Allows for distribution of code without source

• 64-bit bytecodes must co-exist
  • Bytecode version level bumped
  • Forces the image to compile from source on first run
64-bit ENVY Bytecodes

- Subtle impact on loading behavior
  - Cached Bytecodes – Can load code that would otherwise not compile
Contact us

- General information
  - info@instantiations.com
- Sales
  - sales@instantiations.com
- Support
  - support@instantiations.com
- Me
  - sberman@instantiations.com
Thank you for your attention

Questions?