GemTalk Systems

Object and Class Versions In GemStone/S

James Foster
Director of Operations
ESUG
22 August 2016

Agenda

- **Introduction**
- Architecture
- Objects
  - – Lookup (Object Table)
  - – Versions (Commit Records)
- Classes
  - – Lookup (Namespaces)
  - – Versions
- Questions

Limitations of traditional Smalltalks

- Object space (image) must fit in (virtual) RAM
- Object space visible to only one VM
- Sharing objects between VMs is difficult
  - Convert to non-object format (binary, XML, SQL)
  - No built-in object identity for multiple exports
- Object state is lost when VM exits

Welcome to the magical world of GemStone

- Object space limited by disk, not RAM
- Object space shared across multiple VMs on multiple hosts
- Transactional persistence

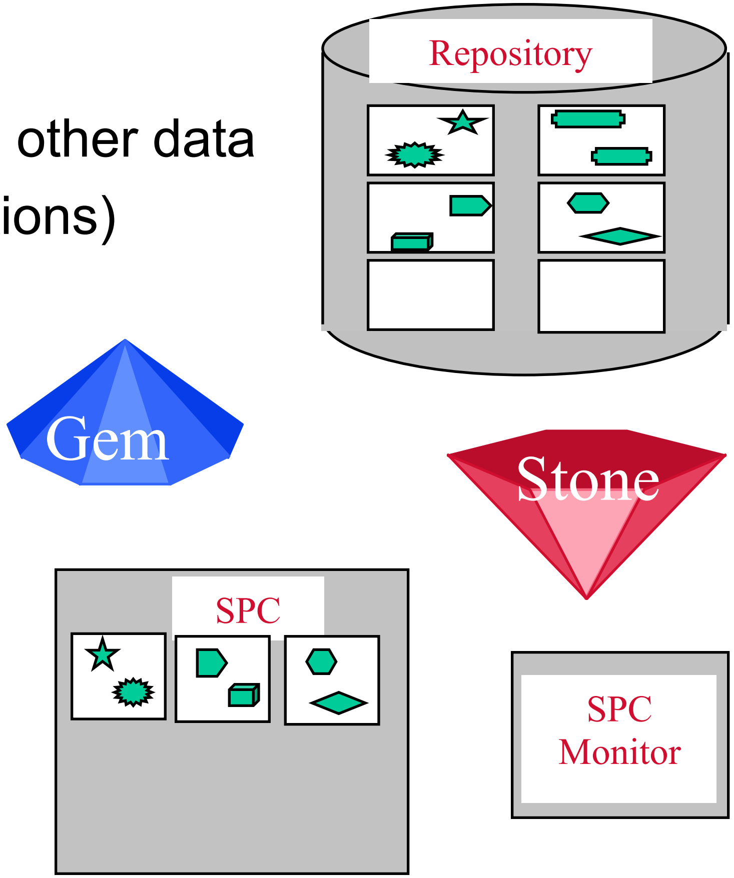
- Image from
  http://www.flickr.com/photos/laffy4k/182219003/
- Creative Commons License
  http://creativecommons.org/licenses/by-sa/2.0/

Agenda

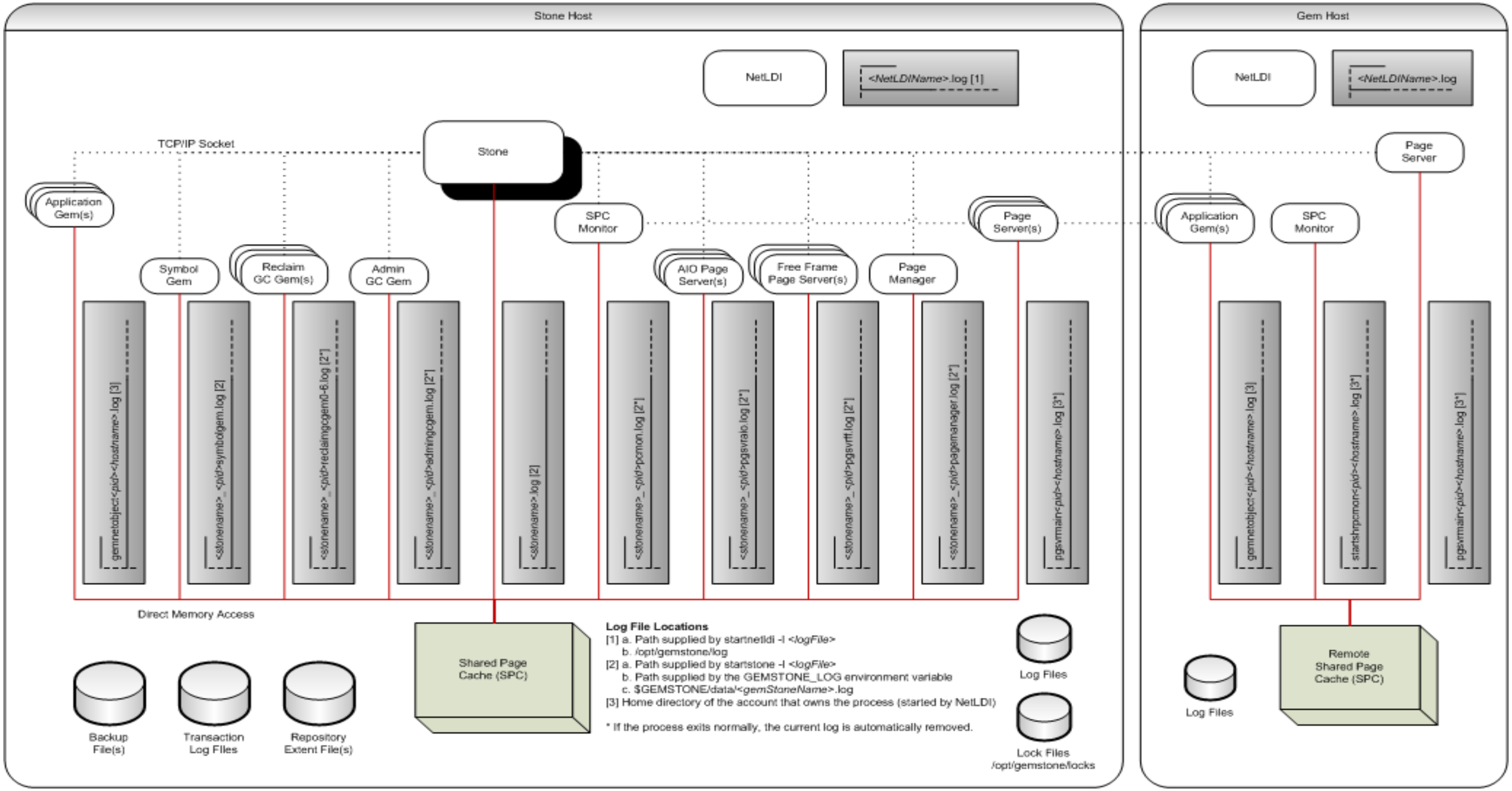
- Introduction
- **Architecture**
- Objects
  - Lookup (Object Table)
  - Versions (Commit Records)
- Classes
  - Lookup (Namespaces)
  - Versions
- Questions

GemStone/S Architecture

- Repository
  - Disk-based “image” holds objects and other data
  - Made up of **extents** (files or raw partitions)
  - Objects are on 16k **pages**
- Gem Process(s)
  - Single Smalltalk virtual machine
- Stone Process
  - Manages concurrency
- Shared Page Cache
  - Fast cache of pages from repository
  - Managed by SPC Monitor process
- Other Processes
  - GC Gems, Symbol Gem, AIO Page Server, Free Frame Server, …



Detailed Component List with Logs



Stone Host
NetLDI
<NetLDIName>.log [1]
TCP/IP Socket
Stone
Application Gem(s)
SPC Monitor
Page Server(s)
Symbol Gem
Reclaim GC Gem(s)
Admin GC Gem
AIO Page Server(s)
Free Frame Page Server(s)
Page Manager
gemnetobject<pid><hostname>.log [3]
<stonename>_<pid>symbolgem.log [2]
<stonename>_<pid>reclaimgcgem0-6.log [2*]
<stonename>_<pid>admingcgem.log [2*]
<stonename>.log [2]
<stonename>_<pid>pcmon.log [2*]
<stonename>_<pid>pgsvraio.log [2*]
<stonename>_<pid>pgsvrff.log [2*]
<stonename>_<pid>pagemanager.log [2*]
pgsvrmain<pid><hostname>.log [3*]
Direct Memory Access
Backup File(s)
Transaction Log Files
Repository Extent File(s)
Shared Page Cache (SPC)
Log Files
Lock Files /opt/gemstone/locks
Gem Host
NetLDI
<NetLDIName>.log
Page Server
Application Gem(s)
SPC Monitor
gemnetobject<pid><hostname>.log [3]
startshrpcmon<pid><hostname>.log [3*]
pgsvrmain<pid><hostname>.log [3*]
Log Files
Remote Shared Page Cache (SPC)

Log File Locations

[1] a. Path supplied by startnetldi -l <*logFile*>
b. /opt/gemstone/log

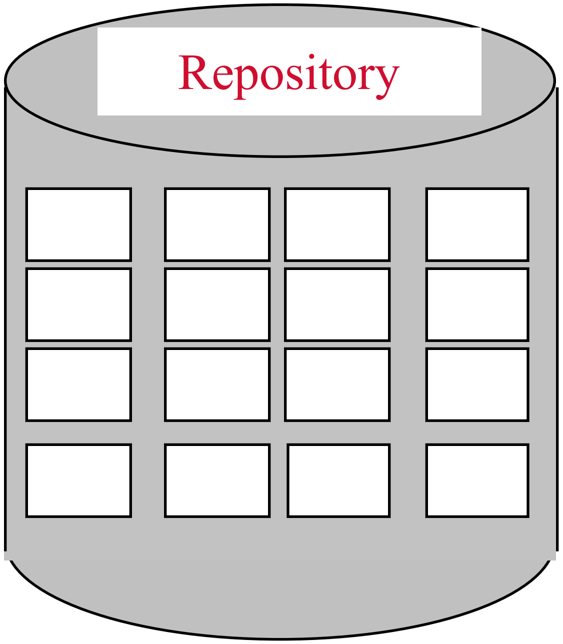
[2] a. Path supplied by startstone -l <*logFile*>
b. Path supplied by the GEMSTONE_LOG environment variable
c. $GEMSTONE/data/<*gemStoneName*>.log

[3] Home directory of the account that owns the process (started by NetLDI)

* If the process exits normally, the current log is automatically removed.

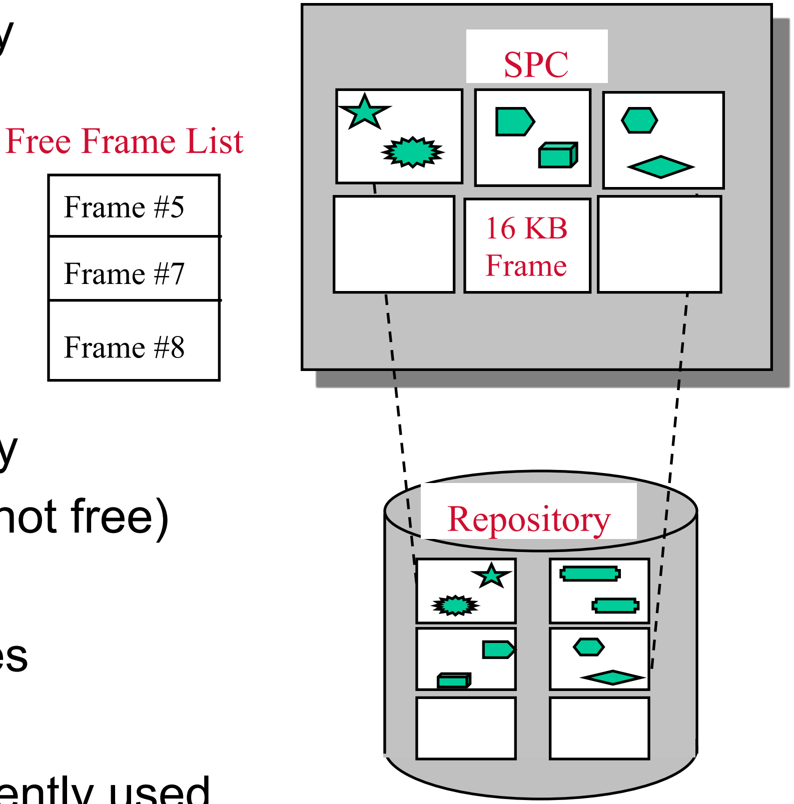
Repository and Extent(s)

- Holds persistent GemStone objects (i.e., the “image”)
- Made up of 1 to 255 *extents* of up to 32 terabytes each
  - On-demand grow
  - Pre-grow to maximum size
- Each extent is composed of 16 KB *pages*
  - Root Pages
  - Object Table Pages
  - Data Pages
  - Commit Record Pages
  - Free OID List
  - Free Page List
- Page ID designates extent and offset



Shared Page Cache

- Typical database challenge: disk is slow
  - In-RAM cache of pages from repository
- Cache is divided into page-sized 'Frames'
  - Frame may be free, clean, or dirty
- Free Frame List
  - Processes take frame(s) from free list
  - Might scan for clean frame (very rare)
- Async IO Page Server(s) write to repository
  - State changes from dirty to clean (but not free)
  - Max 1 per extent
- Free Frame Server(s) scan for clean frames
  - Add to Free Frame List
  - Complex algorithm to identify least recently used



Gem

- Represents a single logged-in database session
- Configurable fixed-size memory allocation
- Executes Smalltalk code
  - Serves most functions of 'Virtual Machine' in other Smalltalks
- Reads objects from Repository
  - Reads pages into SPC, then copies objects to persistent object memory
- Creates and modifies objects
  - New values in temporary object memory
  - Objects are copied to pages in SPC
  - Commit process coordinated with Stone
  - Transaction written to transaction log
  - Pages are written to repository by Stone's AIO Page Servers

Agenda

- Introduction
- Architecture
- Objects
  - **Lookup (Object Table)**
  - Versions (Commit Records)
- Classes
  - Lookup (Namespaces)
  - Versions
- Questions

Object Identifier

- Objects are (generally) referenced by an identifier, *not a* location
  - Object can move (edits, GC compaction)
  - Different versions based on changes in a transaction
    - old values still visible
  - Allows two-way #’become:’ without object-space scan
  - OID (wrongly) called OOP (object-oriented pointer)
  - Exception is possible within a VM

Object Table

- Object Table maps object identifier (OID or OOP) to page identifier
  - Page identifies an extent (file) and offset of 16 KB page
    - Each page has its own object index of offset within page
  - Each new/modified object is on a fresh page
  - Each transaction creates a new OT to point to *current version* of an object

Object Table (OT) - 2

- A new (virtual) Object Table is created by each transaction
  - A database "view" includes a unique OT
  - Optimizations are used to reduce duplication
- Object Table takes space
  - In Repository
  - In Shared Page Cache

Object Lookup

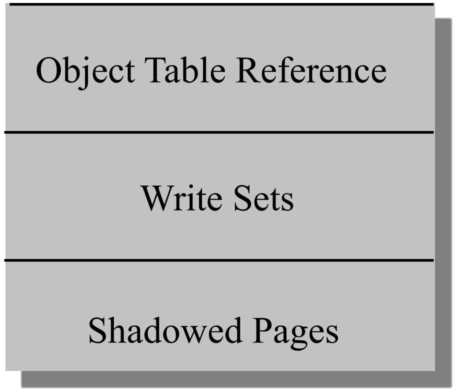
- Reference from an in-Gem-memory object instvar
  - Pointer if already mapped to an in-memory object
  - Otherwise an OID
- Lookup process
  - Check for already-loaded objects (OID to address)
  - Use object-table to determine page
  - Look for page in Shared Page Cache
  - If not present, find a free frame and read from disk
  - Copy from SPC to local Gem memory

Agenda

- Introduction
- Architecture
- Objects
  - Lookup (Object Table)
  - **Versions (Commit Records)**
- Classes
  - Lookup (Namespaces)
  - Versions
- Questions

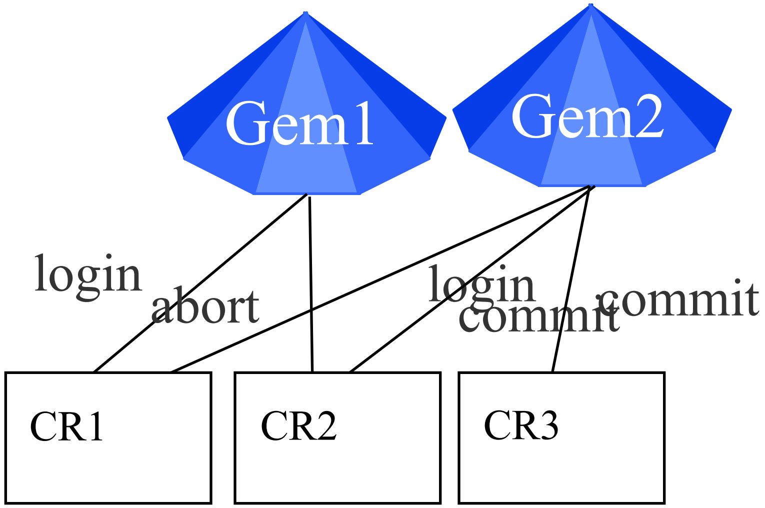
Database View and Commit Record

- On login, Gem has a database view
- Object Table
  - Object ID (OID) == Object Oriented Pointer (OOP)
  - Map to Page (offset in an Extent)
  - Each view is based on a single Object Table
- Each commit creates a *Commit Record*
  - Reference to unique Object Table
  - List of modified objects (*Write Set*)
  - List of Shadowed Pages



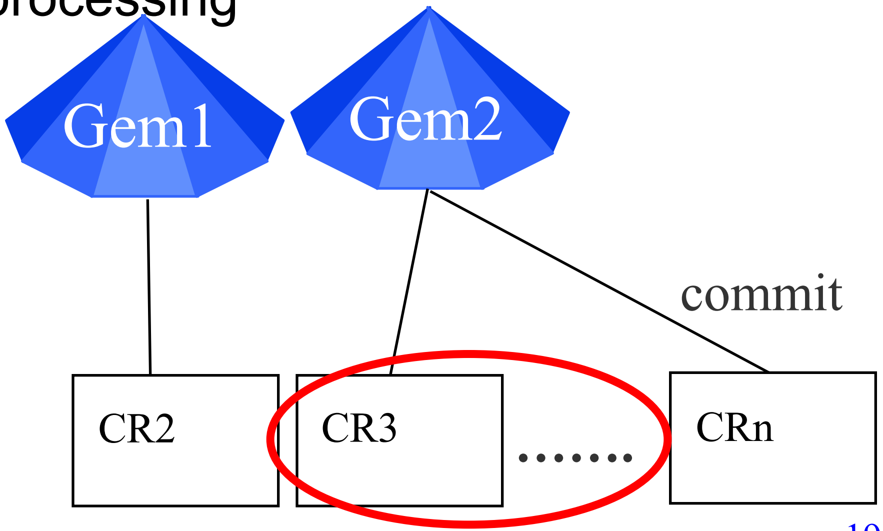
Commit Records

- There is always at least one database view, or Commit Record (CR)
- On login, a Gem is given the most recently created Commit Record
- Other Gems can share the same Commit Record (login or abort)
- Each (non-empty) commit creates a new Commit Record
- An abort moves a Gem to the latest Commit Record
- *Oldest* CR may be disposed of if it is not referenced
- Another commit creates another Commit Record



Commit Record Backlog

- Here we have two Gems and two Commit Records
- Additional commits create more Commit Records (maybe many!)
- Intermediate CRs *cannot* be disposed of if older CR is referenced
  - This can be a major performance issue
- Problems with excess Commit Records
  - They take space in SharedPageCache and/or Repository
  - They slow down new commit processing
  - They delay garbage collection



Agenda

- Introduction
- Architecture
- Objects
  - – Lookup (Object Table)
  - – Versions (Commit Records)
- Classes
  - – **Lookup (Namespaces)**
  - – Versions
- Questions

Typical Smalltalk Name Lookup

- Root: a single Dictionary (subclass) instance
- Lookup at method compile time
  - Compiled method binds to an Association
    - Global name -> object (typically a class name)
- Lookup at runtime
  - (Smalltalk at: #’Array’) new
- Replacing Association value substitutes new object (class)
  - Recompile class’s methods when schema changes
  - References to Association get new value at runtime

GemStone/S Name Lookup: Root

- Root: an instance of **UserProfileSet** (AllUsers)
  - **UserProfile** (name, privileges, groups, symbolList)
  - **SymbolList**: Array of SymbolDictionary instances
  - **SymbolDictionary**: keys constrained to Symbols
- Each user has a unique SymbolList (namespace)
  - SymbolLists may share SymbolDictionaries
    - **Globals** and **Published** are typically shared
    - **UserGlobals** are typically unique

Name Lookup: Compile Time

- Compiler accepts an instance of SymbolList
  - Search SymbolDictionary instances in order
  - Bind to first discovered Association
- Default is user session's *current* symbolList
  - Set at login to symbolList referenced by UserProfile
- Tools (IDE) may provide another SymbolList
  - Effectively a unique Dictionary
  - Opportunity for exploration of tools (packages?)

SymbolLists

- System myUserProfile ==
  GsSession currentSession userProfile
- System myUserProfile symbolList ~~
  GsSession currentSession symbolList
- System myUserProfile symbolList first ==
  GsSession currentSession symbolList first “at login”

Name Lookup: Runtime

- (System myUserProfile symbolList objectNamed: #’Array’) new.
- (Globals at: #’Array’) new.
- (UserGlobals at: #’Array’) new.
  - Could be same or different from one in Globals!

Agenda

- Introduction
- Architecture
- Objects
  - Lookup (Object Table)
  - Versions (Commit Records)
- Classes
  - Lookup (Namespaces)
  - **Versions**
- Questions

Changing a Class Schema

- Traditional Smalltalks do an atomic operation
  - Edit class definition
  - Find and migrate all instances
  - No control over migration process (drop inst vars)
- GemStone does not (usually) migrate instances
  - Repository may be large and finding would be slow
  - Other sessions may have view or edit of instances
  - Modifying session might not have security
- GS Tools may offer immediate migrate instances

Class History (Versions)

- Editing a class schema creates a new class
  - Prior class's methods are compiled into new class
  - New class is added to ClassHistory
  - New class replaces old class in SymbolDictionary
  - Existing instances of old class are unchanged
    - Execute methods in old class!
- ClassHistory
  - Each class references a ClassHistory
  - Each class is referenced by a ClassHistory
  - Provides guidance for migration

Object Migration

- Requires explicit operation
  - – Object>>#’migrate’
  - – Object>>#’migrateFrom:’
  - – Object>>#’migrateFrom:instVarMap:’
  - – Class>>#’migrateInstances:to:’
  - – Class>>#’migrateInstancesTo:’
- Migration creates new instance and swaps identity
  - – Old instance version remain visible from old views
  - – Old instance GCed when no longer visible

Agenda

- Introduction
- Architecture
- Objects
  - Lookup (Object Table)
  - Versions (Commit Records)
- Classes
  - Lookup (Namespaces)
  - Versions
- **Questions**

Questions?

James G. Foster
Director of Operations

GemTalk Systems LLC
15220 NW Greenbrier Pkwy., Suite 240
Beaverton, Oregon, 97006
Voice & Fax: +1 503 766 4714
james.foster@gemtalksystems.com

www.gemtalksystems.com