Accomplishments to Date

* (Feb 10-11) Local dev machine setup.
  + Install MySQL, build a3\_wasteland schema, install extDB (as .dll install), and configure existing mission to use extDB as the persistence framework.
  + Test this configuration and validate that current build is working correctly
* (Feb 12) Establish gitHub repository for g4-gamers missions (repo name: g4-gamers-Wasteland)
* (Feb 13) Incorporate territory persistence prototype (iniDBi only) and extend to support extDB persistence
  + Mission Files modified:
    - \territory\server\monitorTerritories.sqf
    - \territory\server\\territoryPayroll.sqf
    - \territory\client\updateConnectingClients.sqf
    - \A3Wasteland\_settings\main\_config.sqf (add A3W\_territorySaving, A3W\_territoryLogging options)
    - \server\default\_config.sqf (add A3W\_territorySaving, A3W\_territoryLogging options)
  + Mission Files added:
    - \persistence\server\world\tLoad.sqf
    - \persistence\server\world\default\getTerritories.sqf
    - \persistence\server\world\default\saveTerritory.sqf
    - \persistence\server\world\extDB\saveTerritory.sqf \persistence\server\world\extDB\getTerritories.sqf
    - \persistence\server\world\tSaveInit.sqf
  + extDB modifications:
    - add Territory persistence, logging tables to a3\_wasteland\_db schema
      * update extdbModel,mwb to include new tables, relationships & indicies.
      * generate new schema creation .sql as a3wasteland\_db\_v2.04.sql
      * create schema 2.03->2.04 update sql as a3wasteland\_db\_v2.04.sql
    - update \extDB\db\_custom\a3wasteland.ini to include SQL for the following handlers:
      * newTerritory
      * getServerTerritoriesCaptureStatus
      * newTerritoryCaptureStatus
      * updateServerTerritoriesCaptureStatus
* (Feb 15) Initial testing of work so far, debugging tLoad & getTerritories scripts so that getTerritories.sqf will load existing territory data recs from the TerritoryCaptureStatus table, and create new records for territories not existing for the ServerID+MapID combo.
  + Tested and working though extDB call to exec the newTerritoryCaptureStatus sql in getTerritories returns \_markerID as a single element array, instead of the desired integer value. Need to look at the persistence\server\setup\extDB\async\_database.sqf to see whether return val needs to be modified or if I’m calling it incorrectly.
* (Feb 16-17) Tasks:
  + Work on extDB versions *getTerritories.sqf* and *saveTerritory.sqf* to get them functioning correctly
  + *getTerritories.sqf:* loads territory capture persistence info from DB on start-up
    - needed to follow other methods used in other database retrieval method algorithms to:
      * get the existing record set with call to extDB\_Database\_async
      * loop over the record set and save data to a temporary array
      * compare what was returned with defined mission territories and create additional records in the database for any that are missing (this will keep the db in sync if/when mission defined territories are changed)
    - created a3wasteland.ini defined sql ‘*getServerTerritoriesCaptureStatus*’ for use in the extDB\_Database\_async call, using ServerID+MapID as selection keys, and an array of DB column names -> local vars key->value pairs that are used to create a comma-seperated string of columns to retrieve. The result array is then parsed and loaded into a local copy of an array mimicking the global **currentTerritoryDetails** array.
    - This array is then compared to the *config\_territory\_markers* public variable array, and any territories that are missing from the local array are added to the database and to the local array.
      * The key to success here in creating new db records was a two-phase insert/update approach to adding new territory data to the database, and getting the form of the markerName string into the correct form. extDB\_Database\_async does a ‘call compile \_queryResult’ on returning from the extDB extension, so that stuff stored as VARCHAR in the database, needs to saved WITH string literals enclosed in quotes. Checking for a missing territory \_markerName from *config\_territory\_markers* in the database with the *checkServerTerritory* sql call, required formatting the \_markerName string as follows:
        + \_markerName = format ["""%1""", \_x select 0]; , where \_x select 0 is the markerName in *config\_territory\_markers*.
      * If the \_result from the *checkServerTerritory* sql call is false (the rec doesn’t exist), a call to *newTerritoryCaptureStatus* sql returns a newly INSERT’d record ID, that is then updated with the markerName and default data with a call to the *updateTerritoryCaptureStatus* sql which is created and formatted by using an array of [columnName, variableName] mapping pairs that are fed to extDB\_pairsToSQL to generate the correct form of data to feed to the SQL call. The new rec is created in the db, and a record is added to the local temporary territories array.
    - The local territories array is returned to the persistence type independent tLoad caller, which returns it to monitorTerritories where it is assigned to the **currentTerritoryDetails.**
    - **Tested and working as of 2/17/2015 ☺**
  + *saveTerritory.sqf:* Updates territory capture persistence info to DB on capture events
    - compiled in tSaveInit as *fn\_saveTerritory*
    - called from \_handleCapPointTick within monitorTerritories, when a territory capture has occurred. Gets passed an array containing: [\_currentTerritoryID, \_currentTerritoryName, \_newTerritoryOccupiersPlayers, \_currentTerritoryOwner, \_currentTerritoryChrono, \_newCapPointTimer].
    - It converts the list of player objects in the passed \_newTerritoryOccupiersPlayers array to and array of UIDs, and like the other use of the *updateTerritoryCaptureStatus sql,* creates an array of [columnName, variableName] mapping pairs that are fed to extDB\_pairsToSQL to generate the correct form of data to feed to the SQL call to update the db record matching \_currentTerritoryID.
    - **Tested and working as of 2/17/2015 ☺**
  + *territoryPayroll.sqf*: Updates territory capture persistence info with new TimeHeld data
    - calls *fn\_saveTerritory* with [\_territoryId, \_territoryName, \_territoryOccupiers, \_territoryOwner, \_territoryChrono, 0]
    - **Tested and working as of 2/17/2015**
  + *logTerritoryCapture.sqf*: Adds record to the territoryCaptureLog table on territory capture events
    - gets called from \_handleCapPointTick with [\_currentTerritoryID, \_currentTerritoryName, \_newTerritoryOccupiersPlayers, \_currentTerritoryOwner]
    - converts the list of player objects in the passed \_newTerritoryOccupiersPlayers array to and array of UIDs
    - creates an array of [columnName, variableName] mapping pairs that are fed to extDB\_pairsToSQL to generate the correct form of data to feed to the SQL call
    - uses *addTerritoryCaptureLog sql* and the formatted SQL to insert a new record
    - **Tested and working as of 2/17/2015**
  + Issues found:
    - On server restart with OPFOR owned/re-loaded territories, player joins as OPFOR and see’s territory’s owned by OPFOR, but with hatched territory pattern, not solid/transparent pattern. Issue appears to be with how /t/c/updateMarkers is handling data passed from server in it’s remote execVM call to updateMarkers.
      * **Resolution**: This occurs for the first player to join the server, but appears to work okay if that player leaves the server and rejoins.
    - Scores shown in captures on player screen do not reflect the territories held either from previous session, or in current one.
* Feb 18-20 Work: Handle territory capture persistence for Independents
  + Db structure:
    - Add ‘GroupHolder’ (varchar(128)) and ‘GroupHolderUIDs’ (varchar(2048)) columns to territoryCaptureStatus table, and ‘GroupHolder’ (varchar(128) column to territoryCaptureLog table.
  + Global currentTerritoriesDetails array
    - Modify getTerritories.sqf/saveTerritory to/save load new columns in territoryCaptureStatus table to the array as row elements 7&8. Revised currentTerritoriesDetails array is now:

|  |  |  |  |
| --- | --- | --- | --- |
| **Column#** | **Field** | **Data Type** | **Db Column** |
| 0 | Marker ID # | INT | ID |
| 1 | Marker Name | STRING | MarkerName |
| 2 | Players In Area (UIDs) | ARRAY | Occupiers |
| 3 | Players In Area (playerObjects) | ARRAY |  |
| 4 | Current owning team | SIDE | SideHolder |
| 5 | Time held / owned | INT | TimeHeld |
| 6 | Time contested | INT |  |
| 7 | Current owning group (groupObject) | GROUP | GroupHolder (xformed to STRING) |
| 8 | Current owning group (UIDs) | ARRAY | GroupHolderUIDs |

* + - Update getTerritories.sqf to load these new data columns in the global *currentTerritories*
    - Update saveTerritories.sqf to save new db columns in calls from monitorTerritories
  + monitorTerritories:
    - Starting with \_handleCapPointTick, which gets called with [\_territoryOccupiersMapConsolidated, currentTerritoryDetails], where the 1st array is an array of [\_territoryName, [\_player]], and returns a temp/new copy of currentTerritoryDetails which is immediately assigned,
    - \_handleCapPointTick loops over the territories in \_currentTerritoryDetails and for each
      * \_currentTerritoryOccupiersPlayers (col 3, above) is passed to \_teamCountsForPlayerArray, returning [\_teamCounts, \_contested, \_dominantTeam], where \_teamCounts is an array containing [\_team, #players], where \_team and \_dominantTeam can be a SIDE or GROUP object to \_currentTeamCounts. If more than one team/group is in the territory, \_dominantTeam is set to sideUnknown and \_contested is set to true.
      * \_newTerritoryOccupiersPlayers (from \_territoryOccupiersMapConsolidated) is passed to \_teamCountsForPlayerArray, returning [\_teamCounts, \_contested, \_dominantTeam], where \_teamCounts is an array containing [\_team, #players], where \_team and \_dominant can be a SIDE or GROUP object to \_newTeamCounts. If more than one team/group is in the territory, \_dominantTeam is set to sideUnknown. If more than one team/group is in the territory, \_dominantTeam is set to sideUnknown and \_contested is set to true.
      * [\_currentTeamCounts, \_newTeamCounts] is passed to \_handleTeamCounts which returns \_action containing one of [“CAPTURE”,”RESET”, or “BLOCK”]
      * If \_newTeamCounts.\_contested is true or the \_newDominantTeam (from the 2nd call to \_teamCountsForPlayerArray) is not the same as the currentTerritoryOwner:
        + If \_action is “CAPTURE”:

**1st iteration: Broadcast capture to current territory owners (NOTE: NEEDS TEAM/GROUP FIX)**

Increment \_newCapPointTimer

* + - * + If \_action is “RESET”, set \_newCapPointTimer to 0
        + If cap point timer > cap period and this is a new capture:

Set the new marker color from \_newDominantTeam

Reset \_newCapPointTimer

**Call \_onCaptureFinished with [\_currentTerritoryOwner, \_newDominantTeam, \_value, \_currentTerritoryName, \_territoryDescriptiveName]**

Set \_*currentTerritoryOwner* to \_newDominantTeam

**Save new territory status / log territory capture**

**Update player scores for all the players who just capped the territory**

* + - * + Call \_updatePlayerTerritoryActivity with [\_currentTerritoryOwner, \_newTerritoryOccupiersPlayers, \_newDominantTeam, \_action] which just loops over players in \_newTerritoryOccupiersPlayers and updates “TERRITORY\_ACTIVITY” variables on them.
      * Loop over all of the \_newTerritoryOccupiersPlayers, and create up-to-date \_newTerritoryOccupiersUIDs array
      * **Set the \_currentTerritoryData record with [\_currentTerritoryID, \_currentTerritoryName, \_newTerritoryOcupiersUIDs, \_newTerritoryOccupiersPlayers, \_currentTerritoryOwner, \_currentTerritoryChrono, \_newCapPointTimer]**
    - \_handleCapPointTick then returns the new \_currentTerritoryData which is assigned to the global currentTerritoryDetails
    - **The public variable A3W\_currentTerritoryOwners is re-initialized with the array of [territoryName, territoryOwnerTeam] values and rebroadcast**
    - If \_newPlayersWithTerritoryActivity>0, remove them from the \_oldPlayersWithTerritoryActivity array, … at next iter start, those remaining will have the “TERRITORY\_ACTIVITY” variable will be nulled out

Notes:

* Issue with re-joining players not getting back ownership status of previously capped territories on re-connect:
  + Issue arise in the way that /t/c/updateTerritoryMarkers is called.
    - When it is called on a capture (in monitorTerritories.\_onCaptureFinished), pvar\_updateTerritoryMarkers is set to send newly capped territory info to the capturing team, and to everyone else with two different broadcasts:

["pvar\_updateTerritoryMarkers", [\_captureTeam, [[\_captureName], false, \_captureTeam, **true**]]] call fn\_publicVariableAll;

* + - * This calls updateTerritoryMarkers on the target clients with [[\_captureName], false, \_captureTeam, **true**]]
    - ["pvar\_updateTerritoryMarkers", [\_otherTeams, [[\_captureName], false, \_captureTeam, **false**]]] call fn\_publicVariableAll;
      * This calls updateTerritoryMarkers on the target clients with [[\_captureName], false, \_captureTeam, **false**]]
    - This is different than the way that updateTerritoryMarkers is called when a new client connects:

[[[\_markers, true], "territory\client\updateTerritoryMarkers.sqf"], "BIS\_fnc\_execVM", \_player, false] call BIS\_fnc\_MP;

* + - * This passes the array of [\_markerName, \_markerTeam] pairs for all territories for the script to set markers upon.
  + In monitorTerritories.\_onCaptureFinished, when either the team capturing a territory, or the team losing a territory is a group object, the variable “currentTerritories” is set/re-set on the group object, to add/delete the territory being captured/lost, and the group object is passed in the pvar\_ updates being broadcast.

MySQL install and setup:

http://www.mysql.com/why-mysql/windows/

Create the database:

extDB pack with config and stuff:  
<https://github.com/A3Wasteland/Release_Files/raw/master/A3W_extDB_pack.zip>

* Contains a3wasteland\_db\_v2.03.sql for creating the database

1. Extract everything from this ZIP to your Arma 3 install dir

2. Run the a3wasteland db SQL script with your MySQL tool of choice

3. Open extdb-conf.ini and put your MySQL connection infos in the [A3W] ection

4. Try to start your server, and hope it doesn't blow in your face

MySQL Workbench file:  
<https://github.com/A3Wasteland/Release_Files/raw/master/extDB/a3wasteland_db.mwb>

Extdb Setup: Windows

<https://github.com/Torndeco/extdb/releases>

Download the latest Windows.rar for extDB Archive has normal / debug / test versions inside

1. Copy either normal version to /path/to/arma3 i.e(windows/30/\* -> /path/to/arma3)
2. Edit your arma3 launch parameters add @extDB to your mod line.
3. Edit extdb-conf.ini and edit your settings etc...

extDB will only kill Server if config file is missing (from v26+).

**Note:**  
The Debug Version is extra logging, otherwise just use the normal version, this is just for performance reasons. This build is for extDB <https://github.com/Torndeco/extdb>

To setup extDB you need to use the Windows pre-compiled build and edit the extdb-conf.ini Database 2 field with your Database name and login SQL user details to match pretty much your Arma2MySQL one. Make sure the following files are in the root directory of your ARMA 3 server files (where the exe is):

extdb-conf.ini sqlite.db tbb.dll tbbmalloc.dll

You can drag the extdb.dll to that folder as well or you can copy over @extdb over as a regular mod. If you choose to use @extdb as a regular mod your commandline will be: -mod=@life\_server;@extdb

If you are not using @extdb as a mod and choose to drag extdb.dll into the main arma 3 server directory then you only need @life\_server in the commandline.