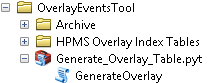
## Introduction

The Custom Overlay Tool provides an easy way to perform table-based overlays between multiple event layers in an ALRS. The output overlay table and feature class can be used as a data product itself, or for performing additional QC, etc.

This tool can handle either a File Geodatabase input, or an SDE input. All inputs for the tool must be in the same database (you can’t choose an ALRS from a FGDB, and a Routes layer from SDE, for example).



**Requirements:**

* Your ArcGIS license level must be elevated to “Advanced” in order to run this tool. Change this level in the ArcGIS Administrator program found under your Start > Programs menu.

### Use Case: HPMS Validations

This tool is highly recommended for the HPMS validation workflow. Some HPMS validations, particularly the coverage validations that compare multiple events, are very difficult to do in Data Reviewer with the Event data directly. Performing an overlay removes that complexity. Flattening out several events into a single overlay table provides the exact format needed to run simple queries in Data Reviewer (and in the Select by Attributes window).

**HPMS Validations:**

* This workflow focuses on Coverage Validations – those validations that compare multiple event layers and are difficult to perform with Data Reviewer
* This workflow does not address simple validations like identifying gaps within a single event layer because there are out of the box tools for performing those validations (GP tools for detecting gaps, Event Editor interactive QC tools, standard Data Reviewer checks for Events, etc.)
* This workflow is not meant to *replace* validations done in UPACS; however, this workflow should help ensure that the data will pass validation in UPACS successfully, thus saving time

### Why a custom tool? Doesn’t ArcMap already do that?

Yes and no. Roads and Highways provides two separate out-of-the-box tools for performing overlays:

1. Linear Referencing Tools > Overlay Route Events
   1. The Good: Uses a tabular method with Route IDs and From/To Measure values to perform overlay
   2. The Bad: Only allows overlaying two events at a time; doesn’t apply a date filter to data
   3. Conclusion: Provides a better output result due to tabular processing
2. Location Referencing Tools > Overlay Route Events
   1. The Bad: Uses a spatial method for performing the overlay, may take a long time to run
   2. The Good: Allows overlaying more than two events at a time
   3. Conclusion: May not provide a very good output result due to spatial processing

The custom Generate Overlay Table tool uses the Linear Referencing > Overlay Route Events tool and its better tabular method for performing the overlay, while also allowing you to configure multiple input event layers. The best of both worlds!

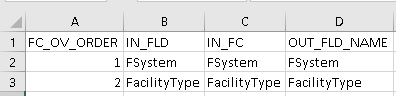
This tool uses a CSV file as an input for configuring the set of events to perform the overlay process on. You can set up any number of CSV files you need (the tool will use one of these at a time). This is a great way to save configurations of the tool, as you’ll very likely have different overlays you’ll need to run on an annual or quarterly basis, depending on the organization’s needs.

## Using the Tool

### Overlay Tool Usage

Several Index Tables (in CSV file format) are pre-configured for use with this tool as an aid to performing HPMS Validations. However, you can create any number of additional separate CSV files as needed. You may decide you have another use case for running overlays between another set of event layers. This information is provided so you can create additional CSV files as needed.

1. Create an Index table with the following column headers:
   1. ‘IN\_FLD’ Type String
   2. ‘IN\_FC’ Type String
   3. ‘OV\_FC\_ORDER’ Type ShortInteger
   4. ‘OUT\_FLD\_NAME’ Type String
   5. For Example:



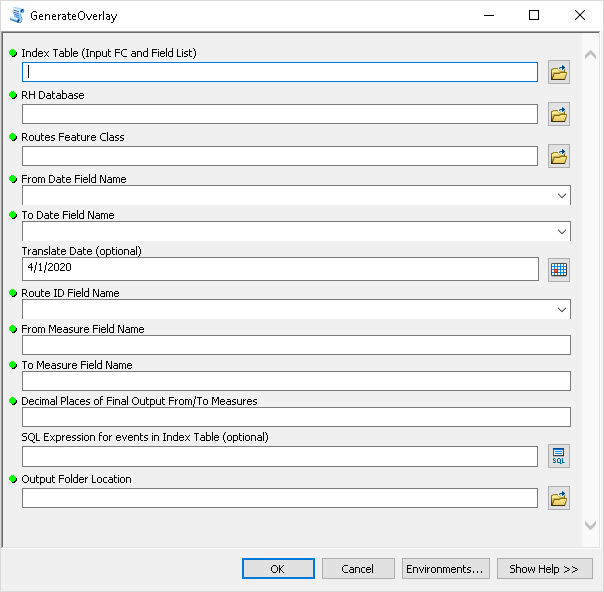
1. Populate the Index Table:
   1. ‘IN\_FLD’ column contains the field names that are going to get pulled from the ALRS feature classes.
   2. ‘IN\_FC’ column contains the ALRS feature class name where the input fields are located.
   3. ‘OV\_FC\_ORDER’ column is populated with the order in which the feature classes are to be overlaid, which determines the order of the fields in the final table.
   4. ‘OUT\_FLD\_NAME’ column contains the output field names for the input fields (i.e. field mapping).

### Running the Overlay Tool

* Navigate in ArcCatalog to the location of “Generate Overlay” toolbox

**NOTE:** Add this path as a fully qualified path name instead of just a letter drive; that will prevent you from having to update the path of the script within the tool.

* Expand the **Generate\_Overlay\_Table.pyt** python toolbox
* Double-click “**GenerateOverlay**”. The following window should pop up:



* Enter Parameters into tool:

1. **Index Table (Field List):** locate the Index Table created (Step 1).
2. **RH Database:** The Database containing the source feature classes and fields needed to make the Overlay Output Table. (One version of the tool can handle the ALRS in SDE, and another version of the tool can handle the ALRS in a File Geodatabase copy; all events used for the tool must be in the same geodatabase).
3. **Routes Feature Class:** The route feature class that is used to map the input event tables. The Route ID, From Measure, and To Measure field names of all input events should match with the corresponding field names in the routes feature class chosen in this parameter.
4. **From Date:** Use the drop-down list to pick the field that contains the From Dates in the Routes feature class. A corresponding field needs to be present and filled out in every input event listed in the Index Table.
5. **To Date:** Use the drop-down list to pick the field that contains the To Dates in the Routes feature class. A corresponding field needs to be present and filled out in every input event listed in the Index Table.
6. **Translate Date (optional):** If you desire to report data from a specific point in time, use the Date picker to choose that date. If you desire the most current data then this field can be left as is.
7. **Route ID Field Name:** Use the drop-down list to pick the field that contains the Route IDs in the Routes feature class. A corresponding field needs to be present and filled out in every input event listed in the Index Table.
8. **From Measure Field Name:** Text string containing the name of the FromMeasure field that will be used for overlays and dissolves (e.g. FromMeasure). The FromMeasure field must be present and uniform in every input event listed in the Index Table.
9. **To Measure Field Name:** Text string containing the name of the ToMeasure field that will be used for overlays and dissolves (e.g. ToMeasure). The ToMeasure field must be present and uniform in every input event listed in the Index Table.
10. **Decimal Places of Final From and To Measures:** Integer value for the number of decimal places the From and ToMeasures of the final output table will be rounded to (e.g. 2; From Measure will be rounded from 32.24312 to 32.24. Note: this rounding operation is a Python function which may truncate numbers, rather than round them. For example, 32.245 may become 32.24 not 32.25)
11. **SQL Expression:** The SQL Query used when copying tables from the RH Database to the Staging Database. Any fields referenced in the query must be located in every input event listed in the Index Table.
12. **Output Folder Location:** Folder directory where the Staging and Output File Geodatabases will be created. You must have read/write permissions for the directory.
13. **Routes Feature Class:** The routes feature class used to when displaying route events. The Route ID must be the same as the Route ID of the input events.

### Tool Output

The tool saves all outputs as well as intermediate processing tables and feature classes in a File Geodatabase. The output File Geodatabase naming convention is:

* OverlayStaging\_[YYYMMDD]\_[HHMMSS].gdb

*For Example:* OverlayStaging\_20190313\_054136.gdb

The final output layer and table are called:

* Feature class: OVERLAY\_OUTPUT\_FC\_[YYYMMDD]\_[HHMMSS]
* Table: OVERLAY\_OUTPUT\_TBL\_[YYYMMDD]\_[HHMMSS]

*For Example:* OVERLAY\_OUTPUT\_FC\_20190313\_054412

*And:* OVERLAY\_OUTPUT\_TBL\_20190313\_054411

### Tool Notes and Limitations

The tool behavior, especially related to Null values, has implications for how queries on the output data must be set up.

* The tool replaces Null values in string type fields are with “.”
  1. Values of “.” In the Output data layer mean that the event has geometry there, but the attribute value is Null
  2. Values of “” or Null in the Output data layer mean that the event does not have any geometry along that stretch of route
* The tool replaces Null values in Double, ShortInteger and LongInteger fields with -999.
  1. In the Overlay Output data layer, a value of -999 in a Double, Short or Long field means there is an event there in the system (there is geometry there for that event layer), but its attribute value is Null
* The tool replaces values of 0 in Double, ShortInteger, and LongInteger fields (except FromMeasure) with -888
  1. In the Overlay Output data layer, a value of 0 in a Double, Short or Long field (other than FromMeasure) means that event layer does not exist along this section of route (there is no geometry there for that event layer)
  2. In the Overlay Output data layer, a value of -888 in a Double, Short or Long field (other than FromMeasure) means there is an event there in the system (there is geometry there for that event layer), and its value is 0 (zero)
* Route ID, From Measure and To measure Fields must be present and uniformly named in all feature classes.
* Source database must be an ALRS database due to the time filter.
* There are two different versions of the tool – one to handle a File Geodatabase input, and one to handle an SDE database input.

## Maintaining the Tool

### Editing the Tool

This information is provided for reference only. The tool is a python toolbox. The tool itself does not need to be edited to set up a new overlay to perform; instead, simply set up a new CSV Index Table file. One reason to edit the actual tool itself would be to improve it so it can use layers from a file geodatabase (and not just the SDE enterprise geodatabase) as inputs for the overlay process. Currently, the tool would fail if the RH Database parameter is pointed to a file geodatabase – the query syntax does not function for both SDE and a file geodatabase. ((> I think this has been done))

1. Navigate in Windows Explorer to location of “Generate Overlay” toolbox
2. Expand the **Generate\_Overlay\_Table.pyt** python toolbox
3. Right-click the “**GenerateOverlay.pyt**” python script > Edit
4. You can also route to the folder using Notepad++ or a similar script editing software and open **GenerateOverlay.pyt** file. If using this method make sure to right-click the **GenerateOverlay.pyt** python toolbox in ArcCatalog and click “Refresh” to update it with the changes before running the tool.

### Tool Methodology

This information is provided for reference. This section details the logic within the tool itself.

1. **Create Staging Geodatabase**
   1. A time stamped Staging GDB is created in the **workspace** parameter.
2. **Copying Tables based on the Index Table**
   1. The environmental workspace is set to the database you have chosen in the **RH Database** parameter
   2. A list of feature classes in the database is created and iterated through.
   3. The name of the feature class is compared with every row of the ‘FC’ column in the index table.
      1. If the feature class name matches with a row in the ‘FC’ column, then the corresponding row in the ‘FIELD’ column is compared with every field name of the matched feature class (e.g. feature class name matches 2nd row of ‘FC’ column then, it then checks if a field name in the matched feature class matches with the 2nd row of the ‘FIELD’ column).
         1. If both the feature class and the field name match then:
            1. A field map is added for that field
            2. a trigger is set so the “Copy Table” tool will run for this feature class.
         2. This process is repeated for the next row in the index table. If multiple fields match for a single feature class then multiple field maps will be made (i.e. can copy multiple fields from the same feature class).
      2. If fields were matched for the feature class then:
         1. The Route ID, From Measure, and To Measure fields are field mapped.
         2. The table from the RH\_SDE Database is copied to the Staging GeoDatabase with only the fields that were field mapped (i.e. Matched fields and Route ID, From Measure, To Measure) and only records that satisfy the SQL Expression parameter.
         3. The Copied table is then cleaned:
            1. From and To Measures are rounded to specified a number of decimal places from the **Decimal Places of Final From and To Measures** parameter;
            2. Null Values are replaced with invalid values based on field type
            3. The table is dissolved based on the field names matched for that feature class
      3. The process is then repeated for the next feature class
3. **Overlaying Copied Tables**
   1. All of the tables from the index table are copied to the Staging Geodatabase.
   2. The MEAN of the ‘INDEX’ column is calculated for each ‘FC’ in the index table and the newly created table is sorted on the MEAN\_INDEX value to determine the order of the overlays for each Feature Class.
   3. The first overlay will be between the first ‘FC’ and the second ‘FC’ value in the sorted table.
   4. The second overlay will be between the third ‘FC’ and the results from the first overlay, with successive overlays mirroring that format.
4. **Formatting Final Tables**
   1. All the copied tables are overlaid in the correct order
   2. The From and To Measures of the final overlay are rounded to the specified number of decimals from the **Decimal Places of Final From and To Measures** parameter;
   3. The final overlay will be dissolved on the ‘FIELD’ list from the index table resulting in a Overlay\_Final\_Disolved table
   4. A Length Calc Field is added to the Overlay\_Final\_Dissolved table and populated by subtracting the From Measure Field from the To Measure Field.
   5. All records with a Length > .001 will be copied from the Overlay\_Final\_Dissolved table to the final output table, located in the Staging Geodatabase.
   6. The route events are displayed using the Routes Feature Class and exported to Feature Class format.

