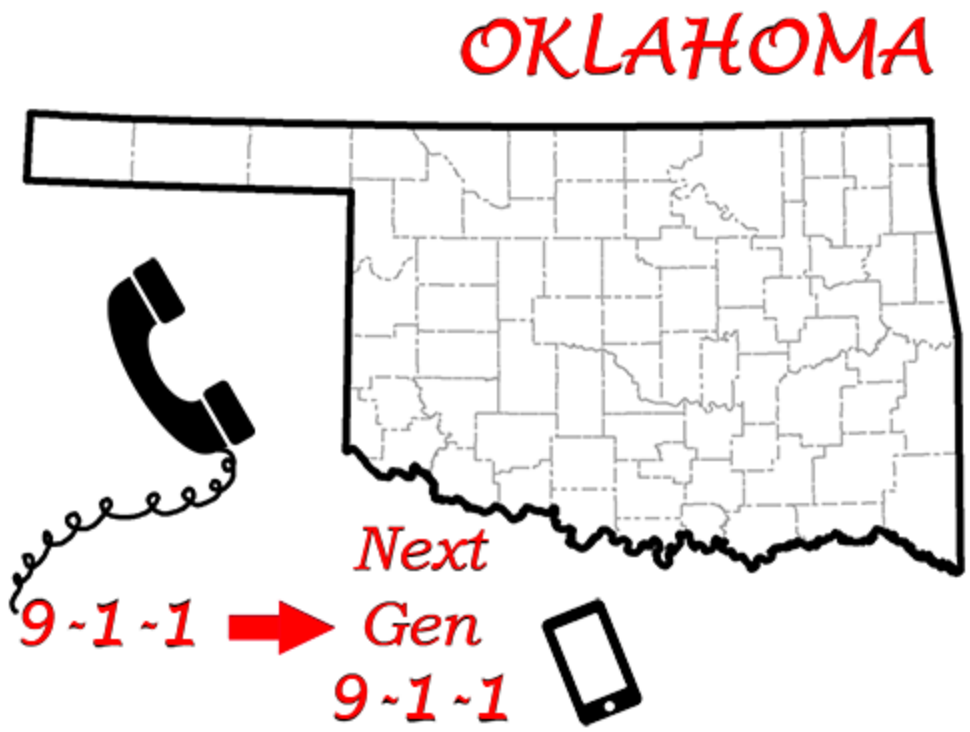
Oklahoma Next-Generation 9-1-1 Pro Toolkit



Abstract

The original NG911 Toolkit was adapted from the Kansas NG911 Toolkit. The most recent Oklahoma Next-Generation 9-1-1 Toolkit has been converted from Desktop to Pro. However, the functionality is the same. The Toolkit is a collection of Tools that identify possible errors and assist the user in complying with the NG911 Oklahoma Standards. Currently, the Toolkit includes 3 toolsets: Prep, Enhancement, and Validation. For specific changes to the Toolkit, see the Change Log and other Toolset Docs. The Toolkit can be found on the Oklahoma Geographic Information Council's standards webpage.

Getting Started

Unzip the NG911 Pro Toolbox and leave all files and folders in the original structure.

Requirements

ArcGIS

This toolkit requires an installation of ArcGIS Pro 3.3. These tools were tested in ArcGIS Pro 3.3. They may

or may not work as expected in other versions of ArcGIS Pro.

Python

ArcGIS Pro 3.3 is installed with Python 3.11. The tools may or may not work with other versions of Python. The `arcpy` and `arcgis` packages, included with ArcGIS, are required, as well as an active ArcGIS license. Standard license is required for Topology, and Advanced license is required for Check Road ESN Advanced. Otherwise, a Basic license is required.

File and Folder Structure

Maintaining file and folder structure is required for proper operation of the Toolkit.

Required folders/files include (but are not limited to) the following:

- `NG911_GIS_Toolkit.pyt`
- `ng911ok` folder and all within
- `config.yml`
- `topology.yml`
- `legacy.yml`
- `counties.yml`
- Files with names like `NG911_GIS_Toolkit_Pro.<...>.xml`

Current Oklahoma NG911 Standards

See the Oklahoma Geographic Information Council's standards webpage for the latest versions of the Oklahoma NG911 and Address Standard.

- Current Standards Version: **3.0**

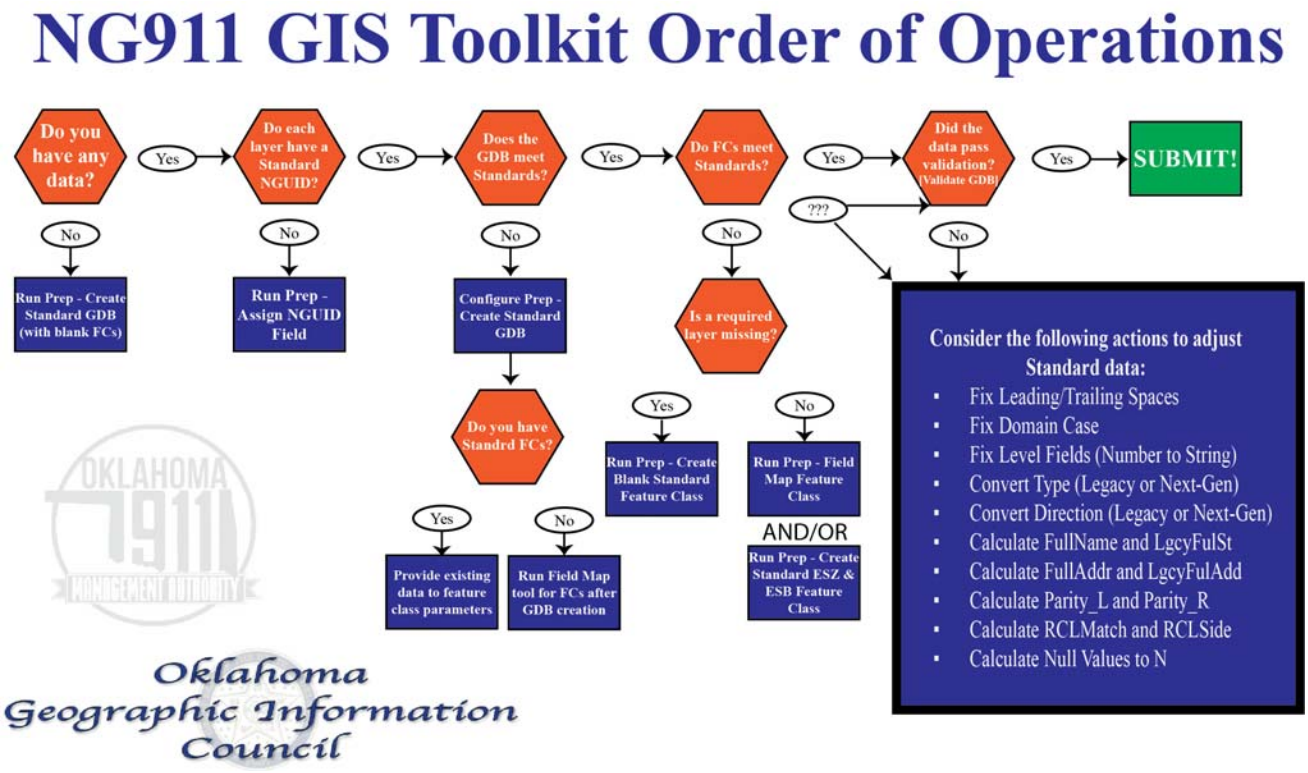
Python API and Toolkit Configuration Documentation

For information about the Python API behind the Toolkit, see the `API Doc` folder or click here.

For information about the YAML configuration files, see API Doc/config_yaml.html.

The API and configuration documentation is not relevant to regular Toolkit usage, but may be useful for those who wish to write custom NG911-related scripts. Please note that editing the YAML configuration files may cause the tools to behave incorrectly or even break completely.

Using the Tools



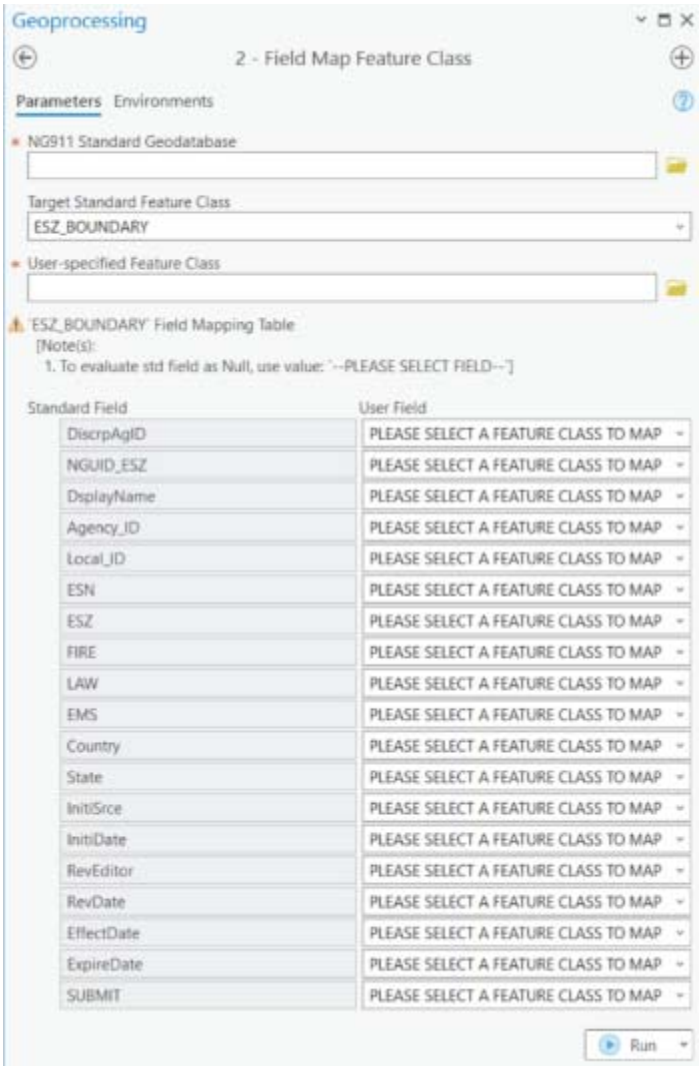
Prep Tools

The Prep tools provide the user with an easy way to create a new standards-compliant geodatabase and convert existing data to comply with the schemas set forth in the Oklahoma NG911 Standards. For more information on where to start as a user, see the Examples document.

Create Empty Geodatabase

To create an empty geodatabase, run the 1 Create Standard GDB tool with a folder location, an output GDB name, a selected coordinate reference system. Leave the feature classes options blank and then 'Run'. The output geodatabase will be in the folder specified and will have a user-specified name. If the user checks the 'Create Blank FC' options, the tool will create an Oklahoma standards-compliant blank feature class when a feature class is not provided **or** when the feature class provided does not match the Oklahoma standards.

Field Mapping



To create a feature class that is correctly formatted, use the 2 Field Map Feature Class tool with the correctly-formatted geodatabase, the user-specified feature class, and the user-specified fields mapped to the appropriate Standard fields. **Triple-check** all user inputs before running the field mapping tool, especially similar fields like `Country` and `County`.

For more information on the individual tools in this toolset, see the Prep Toolset Documentation.

Validation Tools

IMPORTANT: Keep all fields up-to-date, in particular `SUBMIT` and `TopoExcept`. Features where `SUBMIT` is `N` will be omitted entirely from validation checks. Features given a topological exception will be exempted

from given Topology Rules.

The Validate Geodatabase tool in the Validation toolset were written to review the NG911 geodatabase data for submission to the NG911 repository using the Oklahoma NG911 Standards.

For more information on the individual tools in this toolset, see the Validation Toolset Documentation.

Interpreting the Results

This section is pending update.

Enhancement Tools

The Enhancement Tools can be used to automate various tasks involved with data creation and perform quality checks to enhance the data.

Marking for Submission using Fix Submit

The Calculate Null Values to `N` tool will calculate blank/null values in the `SUBMIT` field to `Y` for all the feature classes in a geodatabase. The user would simply mark the features that are not to be included in the submission with `N` and run the Fix Submit tool.

Calculating FullName and LgcyFulSt

The Calculate `FullName` and `LgcyFulSt` tool calculates the `FullName` and/or `LgcyFulSt` fields for the Address Point feature class and/or the Road Centerline feature class.

Converting from Legacy to Next Generation using Fix Street Type and Direction

The Convert Type and Convert Direction tools will calculate the next-gen fields from legacy fields and vice versa.

Calculating the Required Fields: RCLMatch and RCLSide

Calculate `RCLMatch` and `RCLSide` calculates the required fields `RCLMatch` and `RCLSide` in the Address Point feature class.

For more information on the individual tools in this toolset, see the Enhancement Toolset Documentation.

Credits

Scripts written by Riley Baird and Emma Baker with the Oklahoma Transportation Cabinet (OTC)

Last Revised:

April 18, 2025

Disclaimer

The Oklahoma NG9-1-1 GIS Toolbox is provided by the Oklahoma Geographic Information (GI) Council, Oklahoma 9-1-1 Management Authority, Oklahoma Department of Transportation (ODOT), Oklahoma Office of Geographic Information (OGI) , and associated contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the Oklahoma GI Council, Oklahoma 9-1-1 Management Authority, ODOT, OGI, or associated contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

Prep Toolset

Toolset Credits

- Riley Baird, Oklahoma Transportation Cabinet
- Emma Baker, Oklahoma Transportation Cabinet

Last Revised:

July 1, 2025

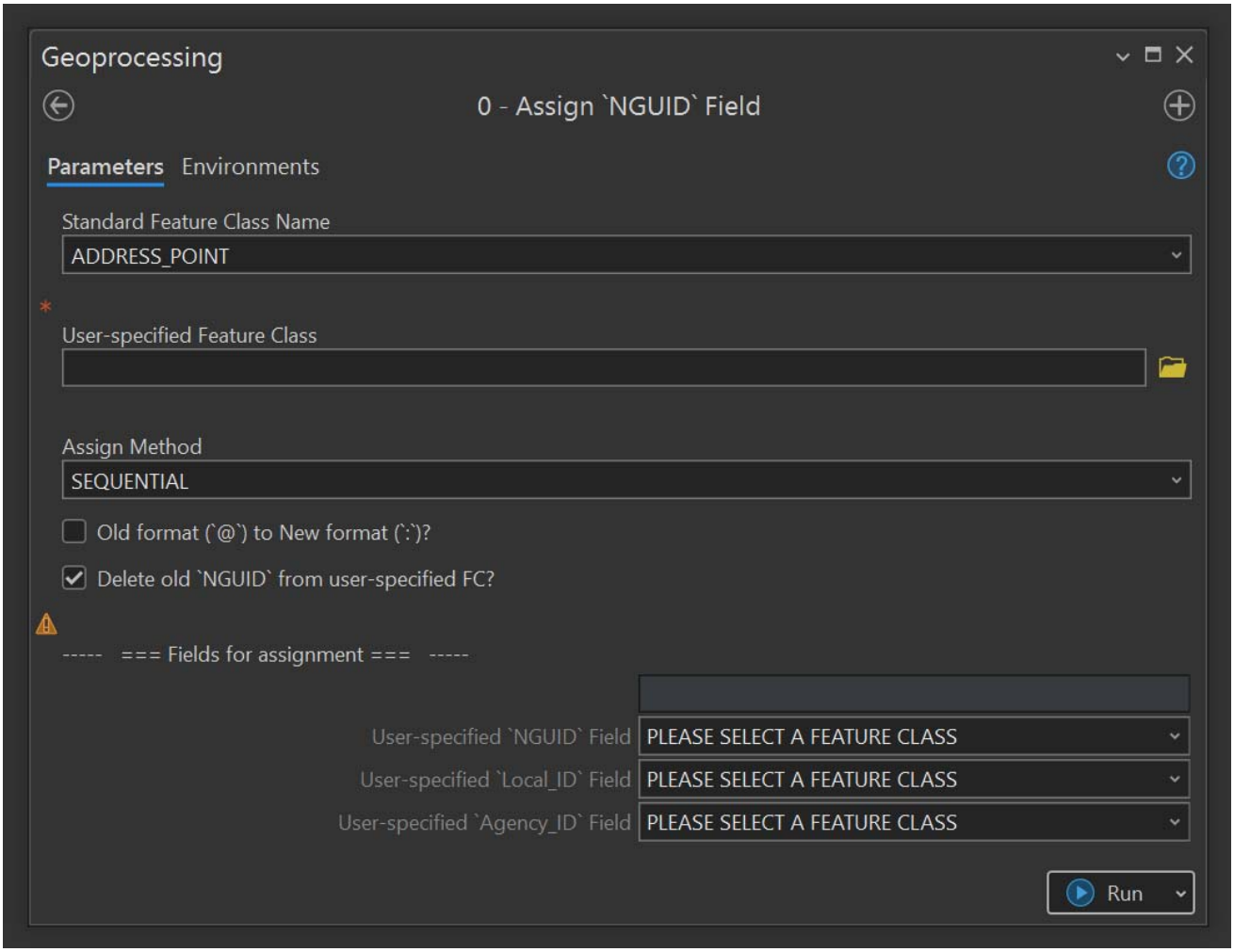
Abstract

The *Prep* tools includes:

- **Assign `NGUID` Field** - A tool that creates a NENA unique `NGUID` field for objects in a non-Standard feature class.
- **Create Standard Geodatabase** - A tool that creates a new user-named Standard geodatabase with the correct schema in the specified folder location using a selected coordinate reference system. These tools should be run to ensure the feature classes and geodatabase that will be submitted are formatted with the correct domains and field names.
- **Field Map Feature Class** - A tool that creates field maps from user-specified feature class fields to Standard feature class fields with the correct schema including names, types, lengths, and domains.
- **Create Blank Standard Feature Class** - A tool that creates blank Standard feature classes in a Standard geodatabase.
- **Create Standard ESZ & ESB Feature Classes** - A tool that dissolves a ESB feature class to create Standard ESZ and ESB Feature Classes.

Tools

0 - Assign `NGUID` Field



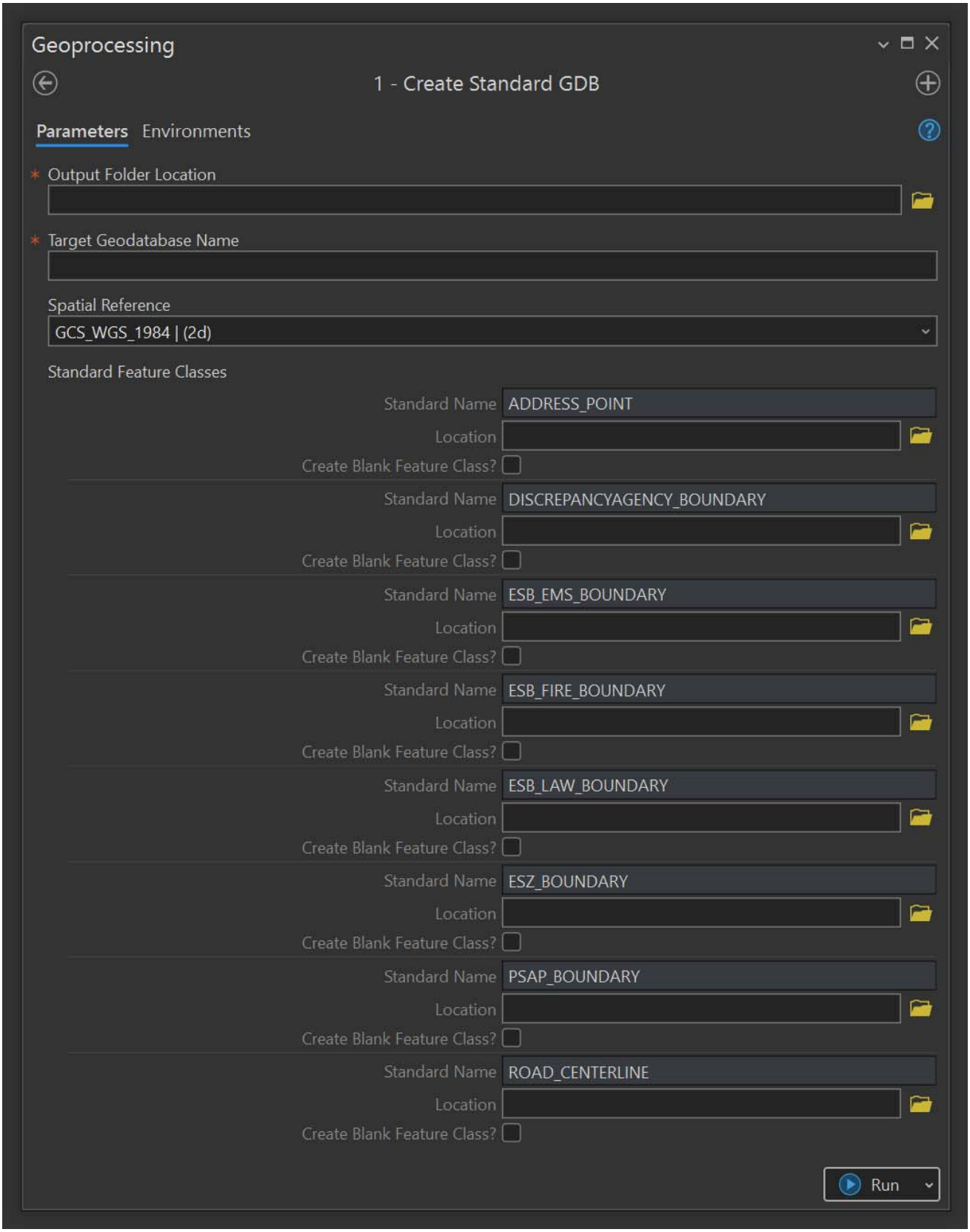
- The tool creates a NENA unique `NGUID` for features in a feature class. The tool concatenates user-specified `URN`, `Local`, or `Agency` fields or reformats user-specified unique NGUID field for user-specified feature class. The feature class does **NOT** have to be Standards-compliant. The tool will create a new Standard-compliant `NGUID` field and calculate it using user-specified parameters. The user-specified assign method will determine how the new field is concatenated. Certain assign methods require certain fields.

Usage

1. Select the Standard feature class name.
2. Select feature class to assign `NGUID`.
3. Select assign method to specify which `NGUID` creation method to use.
    - 'NGUID' - Requires user-specified `NGUID` field. Method required for a format change.
    - 'LOCAL' - Requires `Local` field.
    - 'SEQUENTIAL' - Uses a sequential count for the Local portion of the `NGUID` field
4. Optional switch for Old format (`@`) to New format (`:`).
5. Optional switch to delete old `NGUID` field. Since the tool will create a new `NGUID` field to ensure that the Standard schema is followed, the user may specify whether to keep or delete the old `NGUID` field.

6. Optional name for the old `NGUID` field that will be used if user selects to not delete the old field (usage 5).
7. Field table containing field options for the `NGUID`, `Local`, and `Agency` fields. The filter list is based on the user-specified feature class (usage 2).
8. Execute the tool.

1 - Create Standard GDB

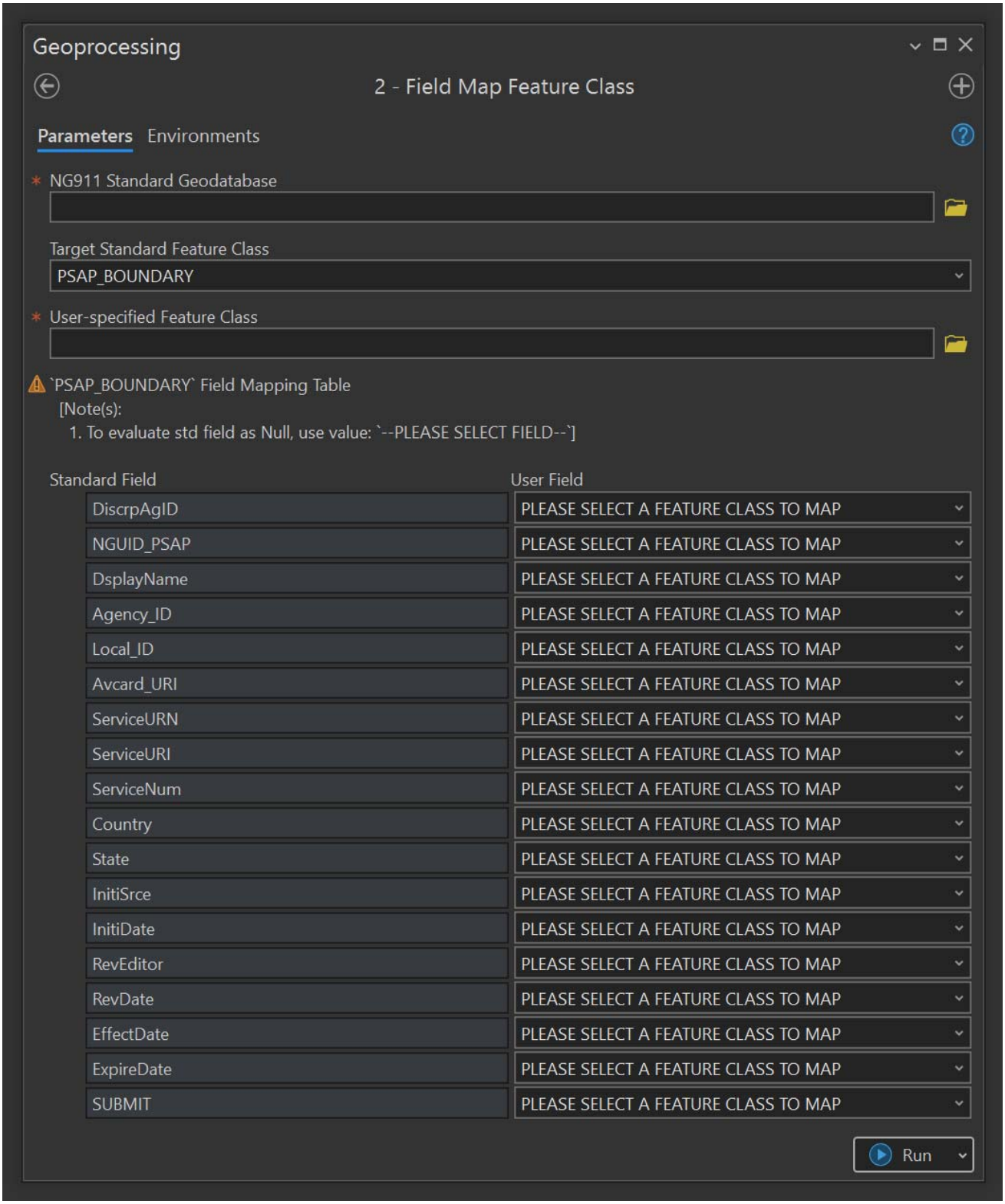


- The tool creates a Standard geodatabase in the specified folder location using a selected coordinate reference system. None, some, or all feature classes may be provided, but they must follow the Standard schema. Blank Standard feature classes can also be created in the Standard geodatasbase.

Usage

1. Select folder location.
2. Select geodatabase name (without '.gdb'.).
3. Select Spatial Reference option. If '--USER SPECIFIED--' is selected, the user will need to select a Spatial Reference manually.
4. Select Standard feature classes or the option to create blank feature classes. If nothing is provided, nothing will be created.
5. Execute the tool.

2 - Field Map Feature Class

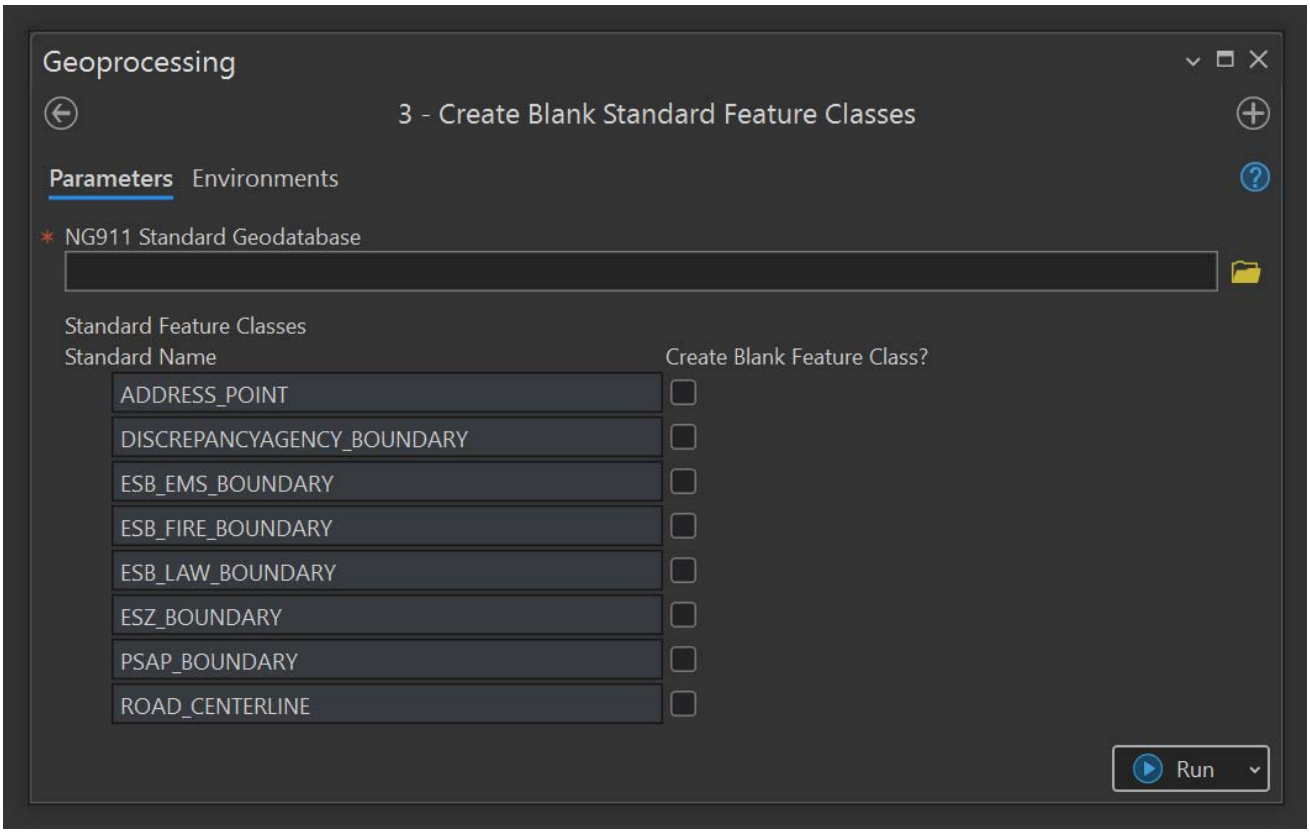


- A tool creates field maps from user-specified feature class fields to Standard feature class fields with the correct schema including names, types, lengths, and domains. An empty Standard feature class is first created and then the Standard-compliant fields are added. Field mapping between the user fields and the Standard fields is performed. The values in the user fields are assessed and modified if necessary based on the Standards.

Usage

1. Select Standard geodatabase.
2. Select Standard feature class name.
3. Select user-specified feature class.
4. Select fields to be mapped. If no field is specified, the field will remain blank.
5. Execute the tool.

3 - Create Blank Standard Feature Class

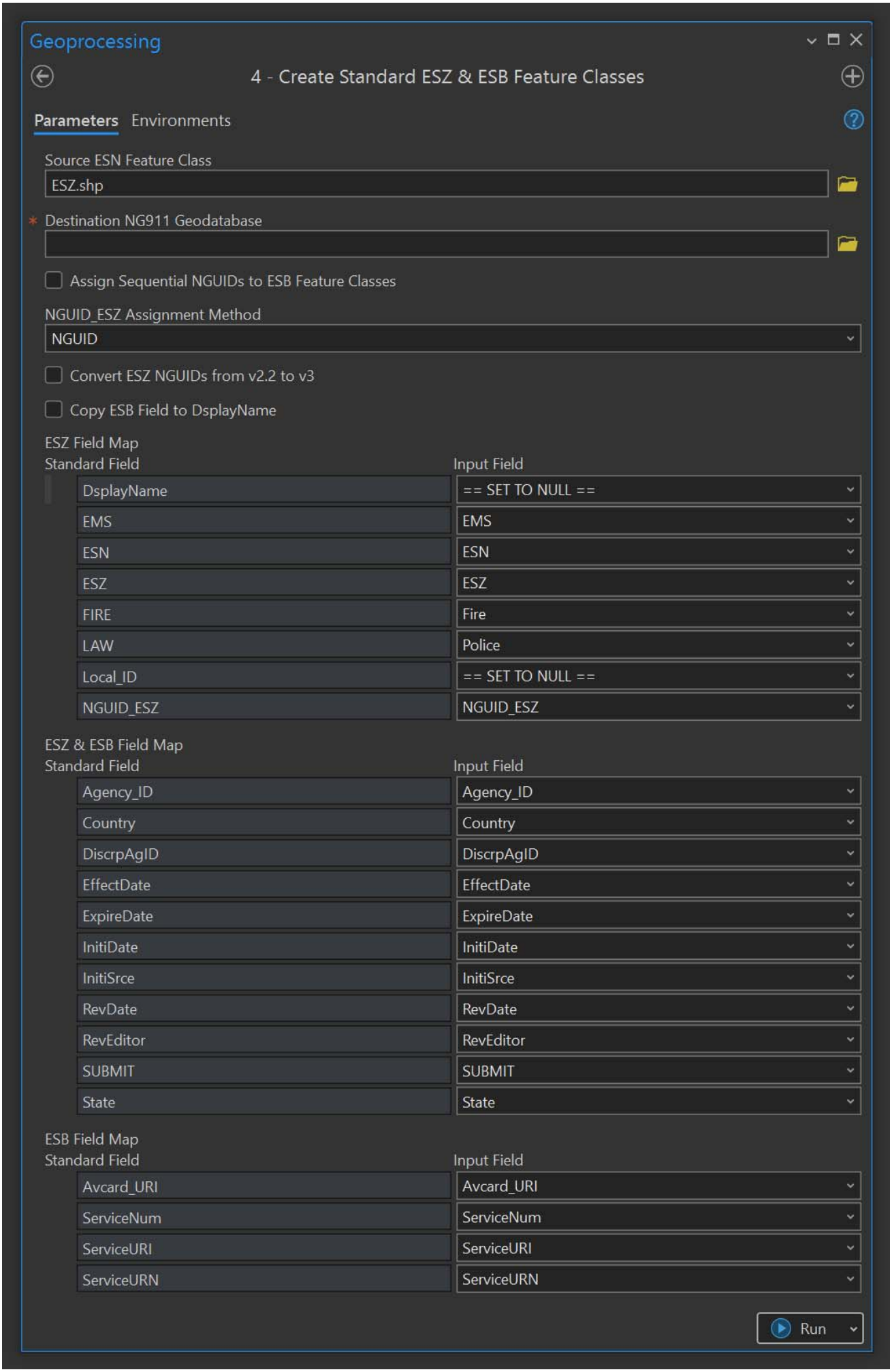


- A tool that creates blank Standard feature classes in a Standard geodatabase.

Usage

1. Select Standard geodatabase.
2. Select the Standard feature class names to create blank.
3. Execute the tool.

4 - Create Standard ESZ & ESB Feature Classes


Geoprocessing
4 - Create Standard ESZ & ESB Feature Classes
Parameters
Environments
Source ESN Feature Class
ESZ.shp
* Destination NG911 Geodatabase
Assign Sequential NGUIDs to ESB Feature Classes
NGUID_ESZ Assignment Method
NGUID
Convert ESZ NGUIDs from v2.2 to v3
Copy ESB Field to DsplayName
ESZ Field Map
Standard Field
Input Field
DsplayName
== SET TO NULL ==
EMS
EMS
ESN
ESN
ESZ
ESZ
FIRE
Fire
LAW
Police
Local_ID
== SET TO NULL ==
NGUID_ESZ
NGUID_ESZ
ESZ & ESB Field Map
Standard Field
Input Field
Agency_ID
Agency_ID
Country
Country
DiscrpAgID
DiscrpAgID
EffectDate
EffectDate
ExpireDate
ExpireDate
InitiDate
InitiDate
InitiSrce
InitiSrce
RevDate
RevDate
RevEditor
RevEditor
SUBMIT
SUBMIT
State
State
ESB Field Map
Standard Field
Input Field
Avcard_URI
Avcard_URI
ServiceNum
ServiceNum
ServiceURI
ServiceURI
ServiceURN
ServiceURN
Run

- This tool takes dissolves an ESZ feature class and creates the Standard ESZ_BOUNDARY, ESB_EMS_BOUNDARY, ESB_LAW_BOUNDARY, and ESB_FIRE_BOUNDARY feature classes in the Standard geodatabase.

Usage

1. Select ESZ feature class to dissolve.
    - Must have, at minimum, fields that map to `ESZ`, `EMS`, `FIRE`, and `LAW`.
2. Select Standard geodatabase.
3. If desired, select the option to assign sequential NGUIDs to ESB feature classes.
    - This will have no effect on the output ESZ feature class, only the output ESB feature classes.
4. Select NGUID assignment method for `NGUID_ESZ`.
    - This will not have any effect on the output ESB feature classes, only the output ESZ feature class.
    - Depending on the method, fields that map to one or more of `NGUID_ESZ`, `Agency_ID`, and/or `Local_ID`.
    - If the "NGUID" method is selected, an option will be available to convert existing v2.2 NGUIDs to v3 NGUIDs.
5. If desired, select the option to copy the ESB fields to the `DsplayName` fields of the respective feature classes.
    - This will have no effect on the output ESZ feature class, only the output ESB feature classes.
    - For example the source field that is set to map to `EMS` will be mapped to the `DsplayName` field of ESB_EMS_BOUNDARY.
6. Select the fields to map.
7. Execute the tool.

Support Contact

For issues or questions, please contact through email Riley Baird at rbaird@odot.org or Emma Baker at ebaker@odot.org with the Oklahoma Transportation Cabinet, and please include in the email which script you were running, any error messages, and a zipped copy of your geodatabase. Change the file extension from `zip` to `piz` so it gets through the email server. If there are further data transfer issues, contact Emma or Riley to make alternative data transfer arrangements.

Disclaimer

The Oklahoma NG9-1-1 GIS Toolbox is provided by the Oklahoma Geographic Information (GI) Council, Oklahoma 9-1-1 Management Authority, Oklahoma Department of Transportation (ODOT), Oklahoma Office of Geographic Information (OGI) , and associated contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a

particular purpose are disclaimed. In no event shall the Oklahoma GI Council, Oklahoma 9-1-1 Management Authority, ODOT, OGI, or associated contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

Enhancement Toolset

Toolset Credits

- Riley Baird, Oklahoma Transportation Cabinet
- Emma Baker, Oklahoma Transportation Cabinet

Last Revised:

January 09, 2025

Abstract

These tools automate various tasks involved with data creation and/or modifications.

Tools

Calculate/Convert/Fix Field(s)

Calculate/Convert/Fix

- Tool calculates, converts, or fixes fields in Standard feature classes. More details provided below.

Usage

1. Select Standard geodatabase.
2. Select analysis.
3. Select required and/or optional parameters (see below for more information).
4. Execute the tool.

Calculate/Convert/Fix Field(s) - Details

Common Across Tool

Street Fields

- Fields include `PreMod`, `PreDir`, `PreType`, `PreTypeSep`, `Street`, `StreetType`, `SufDir`, and `SufMod`.

Next-Gen Fields

- Fields include `Street`, `PreTyp`, `StreetType`, `PreDir` and `SufDir`.

Legacy Fields

- Fields include `LgcyStreet`, `LgcyPreTyp`, `LgcyType`, `LgcyPreDir` and `LgcySufDir`.

Side for Roads

- Options include `LEFT` ('L') and `RIGHT` ('R')

Variant Options

- 'NEXT-GEN' - Use and/or Calculate next-gen fields
- 'LEGACY' - Use and/or Calculate legacy fields

Calculate `FullName` and `LgcyFulSt`

- Calculates the `FullName` and `LgcyFulSt` fielda of the Address Point Standard feature class and/or the Road Centerline Standard feature class. For the `FullName` fields, the street fields are used with next-gen fields, and the legacy fields are replaced with the Legacy version of the fields for the `LgcyFulSt` field.

Usage (con't.)

- Select at least one of the Standard feature class names for analysis.
- Select at least one of the variant options for analysis.

Calculate `FullAddr` and `LgcyFulAdd`

- Calculates the `FullAddr` fand `LgcyFulAdd` fielda of the Address Point Standard feature class and/or the Road Centerline Standard feature class. For the `FullAddr` fields, the `AddNumPre`, `AddNumber`, `AddNumSuf`, plus the street fields are used, and the [legacy direction fields] and [legacy type fields] are preplaced with the Legacy version of the fields for the `LgcyFulAdd` field.

Usage (con't.)

- Select at least one of the variant options for analysis.

Calculate `Parity_L` and `Parity_R`

- Calculates the `Parity_side` fields of a road centerline feature class. The fields used for the

calculation are: Add_side_From and Add_side_To. The tool will used the first and last digit on both sides to determine whether the number is EVEN, ODD, BOTH, or ZERO.

Usage (con't.)

- N/A

Calculate Null Values to N

- Calculates null values for fields with YESNO domain as N.

Usage (con't.)

- Select at least one of the Standard feature class names for analysis.

Calculate RCLMatch and RCLSide

- Requires the RCLMatch field. First, dataframes are created for the Address Point and Road Centerline featuer classes. Next, the street fields concatenated strings for each feature class. Then, the Road Centerline dataframe is joined to the Address Point dataframe using the street fields concatenated string. Non-matched address objects are then removed from the joined dataframe. Next, the AddNumber field value is then compare to the range of Add_side_From value to Add_side_To value with the Partiy_side used to determine the step number; this will determine the side of the road (RCLSide). Non-matches are then removed from the joined dataframe. The PreNum value is then compare the both Add_side_Pre value, and the MSAGComm field value is compared to the MSAGComm_side value. Once these comparisons are complete, the possible matches are returned. If only one match is returned, the NGUID_RDCL value is returned as RCLMatch. Values that results with ties start with the string TIES with the NGUID_RDCL values returned as a pipe-separated local:agency string. RCLSide is set to NO MATCH The rest of the non-matched values are populated as NO MATCH.

Usage (con't.)

- N/A

Calculate MSAGComm

- Calculate MSAGComm field for Address Point feature class and MSAGComm_side fields for Road Centerline feature class.

Usage (con't.)

- Select at least one of the Standard feature class names for analysis.

Convert Type (Legacy or Next-Gen)

- Convert Type next-gen fields/legacy fields based on the variant options. The Legacy option calculates the legacy fields from the next-gen fields, while the Next-Gen option calculates the next-gen fields from the legacy fields.

Usage (con't.)

- Select at least one of the Standard feature class names for analysis.
- Select one of the variant options for analysis.

Convert Direction (Legacy or Next-Gen)

- Convert Direction next-gen fields/legacy fields based on the variant options. The Legacy option calculates the legacy fields from the next-gen fields, while the Next-Gen option calculates the next-gen fields from the legacy fields.

Usage (con't.)

- Select at least one of the Standard feature class names for analysis.
- Select one of the variant options for analysis.

Fix Leading/Trailing Spaces

- Fix leading/trailing spaces in fields with TEXT type.

Usage (con't.)

- Select at least one of the Standard feature class names for analysis.

Fix Domain Case

- Fix values that match values in domain.

Usage (con't.)

- Select at least one of the Standard feature class names for analysis.

Fix Level Fields

- Fix FromLevel and/or ToLevel fields from number-version to text-version to match domain in Road Centerline feature class.

Usage (con't.)

- Select at least one of the Standard field names for analysis.

Support Contact

For issues or questions, please contact through email Riley Baird at rbaird@odot.org or Emma Baker at ebaker@odot.org with the Oklahoma Transportation Cabinet, and please include in the email which script you were running, any error messages, and a zipped copy of your geodatabase. Change the file extension from `zip` to `piz` so it gets through the email server. If there are further data transfer issues, contact Emma or Riley to make alternative data transfer arrangements.

Disclaimer

The Oklahoma NG9-1-1 GIS Toolbox is provided by the Oklahoma Geographic Information (GI) Council, Oklahoma 9-1-1 Management Authority, Oklahoma Department of Transportation (ODOT), Oklahoma Office of Geographic Information (OGI) , and associated contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the Oklahoma GI Council, Oklahoma 9-1-1 Management Authority, ODOT, OGI, or associated contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

Validation Toolset

Toolset Credits

- Riley Baird, Oklahoma Transportation Cabinet
- Emma Baker, Oklahoma Transportation Cabinet

Last Revised:

March 17, 2025

Abstract

The data validation tools perform a variety of basic verification checks against the NG911 Data Model template to determine if the data is ready for submission. The scripts are organized to validate data by specific layers or groups of layers, and multiple optional tests are included for each set. Any issues found with the data will be reported in tables added to the geodatabase as an "Error" or a "Notice". Notices will not prevent submission.

The scripts can be run multiple times as necessary so users can correct basic issues prior to submitting their NG911 updates. Currently, these data validation tools do not provide complete quality assurance (QA) of the data.

Be advised that most of these checks do not *correct* any errors, only *make note* of them.

Methodology is expanded below for the following validation checks: **Check Feature Locations**, **Check Topology**, **Check RCLMatch**, **Check Frequency**, **Check ESN and Muni Attributes**, and **Find Overlaps**.

Tools

A total of nine tools exist in the Validation Toolset. The individual tools and their available checks are described below. Nested bullet points indicate checks that are automatically run when the option at the top-level bullet point is selected.

1 Check Template

- **Check Layer List:** Checks for required layers. Missing required layers are reported in the *TemplateCheckResults* table. This also checks for a feature dataset named `NG911` and checks that it uses the correct spatial reference.

- **Check Required Fields:** Checks feature classes to ensure that they contain the required fields.
- **Check Required Field Values:** Returns errors if required fields contain null values.
- **Check Submission Counts:** Checks to see how many features are marked for submission.
- **Find Invalid Geometry:** Returns errors if points, lines, or polygons have fewer than 1, 2, or 3 points, respectively.
- **Check GDB Domains:** Returns errors if geodatabase Domains don't match the Oklahoma standards.

2 Check Address Points

- **Check Values Against Domains:**
    - Ensures that fields with domains contain values that match the domain.
- **Check MSAGComm Spaces:** Checks for and removes leading or trailing spaces in `MSAGComm` fields.
- **Check `RCLMatch`:** (if it exists) Validates the `RCLMatch` field of the ADDRESS_POINT feature class against the ROAD_CENTERLINE feature class.
- **Check Feature Locations:** Compares features in a feature class to the discrepancy agency boundary to ensure that they lie within a boundary.
- **Check Address Point Frequency:** Checks for identical address records and returns errors if any are found.
- **Check Unique ID:** Ensures that no address records share unique IDs and unique IDs are formatted properly.
- **Format:** Ensures that are unique IDs for address records are formatted properly. Example: `LAYERNAME_{ABC-123}@AGENCYID`
- **Frequency:** Ensures that no address records share unique IDs.
- **Check ESN and Muni Attributes:** Selects address points using ESN or muni polygons to check `ESN` and `City` attributes match those of the polygons.
- **Check Next-Gen Against Legacy Fields:** Ensures the fully-written-out Next-Gen field values match the abbreviated values in the corresponding Legacy fields. Example: Record with `StreetType` attribute of `AVENUE` must have `LgcyType` attribute of `AVE`.

3 Check Roads

- **Check Values Against Domains:**
- Ensures that fields with domains contain values that match the domain.
- **Check MSAGComm Spaces:** Checks for and removes leading or trailing spaces in `MSAGComm` fields.
- **Check Parities:** Returns a notice if address ranges do not match parity fields
- **Check Feature Locations:** Compares features in a feature class to the discrepancy agency boundary to ensure that they lie within a boundary.
- **Check Unique ID:** Ensures that no road centerline records share unique IDs and unique IDs are formatted properly.
- **Format:** Ensures that are unique IDs for road centerline records are formatted properly. Example: `LAYERNAME_{ABC-123}@AGENCYID`

- **Frequency:** Ensures that no road centerline records share unique IDs.
- **Check for Cutbacks:** Returns a notice if a road centerline feature contains angle on the open interval (0, 55) degrees, which could potentially indicate a cutback.
- **Check Directionality:**
- Ensures that address ranges run from low to high numbers.
- **Check Next-Gen Against Legacy Fields:** Ensures the fully-written-out Next-Gen field values match the abbreviated values in the corresponding Legacy fields. Example: Record with `StreetType` attribute of `AVENUE` must have `LgcyType` attribute of `AVE`.
- **Check Overlapping Address Ranges:** Checks for numerical overlaps in address ranges between road centerline features.

4 Check Boundaries

- **Check Values Against Domains:** Ensures that fields with domains contain values that match the domain.
- **Check Feature Locations:** Compares features in a feature class to the discrepancy agency boundary to ensure that they lie within a boundary.
- **Check Unique ID:** Ensures that no boundary records share unique IDs and unique IDs are formatted properly.
- **Format:** Ensures that are unique IDs for boundary records are formatted properly. Example: `LAYERNAME_{ABC-123}@AGENCYID`
- **Frequency:** Ensures that no boundary records share unique IDs.

5 Check Additional Layers

- **Check Values Against Domains:** Ensures that fields with domains contain values that match the domain.
- **Check Feature Locations:** Compares features in a feature class to the discrepancy agency boundary to ensure that they lie within a boundary.
- **Check Unique ID:** Ensures that no other records share unique IDs and unique IDs are formatted properly.
- **Format:** Ensures that are unique IDs for other records are formatted properly. Example: `LAYERNAME_{ABC-123}@AGENCYID`
- **Frequency:** Ensures that no other records share unique IDs.

6 Clear Results Table (Optional)

- **Clear Template Results:** Deletes all records in table *TemplateCheckResults*.
- **Clear Field Value Results:** Deletes all records in table *FieldValuesCheckResults*.

7 Verify Topology Exceptions (Optional)

- Double-checks that all road centerline topology error are recorded as exceptions in the data and

the topology.

7.1 ESB Gap Locations (Optional)

- Performs a Symmetrical Difference to find gaps between the ESB layers and the Discrepancy Agency Boundary layer. Requires an advanced license. Required layers include ESB_LAW_BOUNDARY, ESB_FIRE_BOUNDARY, ESB_EMS_BOUNDARY, and DISCREPANCYAGENCY_BOUNDARY.

8 Check All Required

- **Template Checks**
- Deletes existing result tables
- Check Spatial Reference
- Check Layer List
- Check GDB Domains
- Check Required Fields
- Check Required Field Values
- Check Submission Numbers
- Find Invalid Geometry
- **Common Layers Checks**
- Check Values Against Domain
- Check Feature Locations
- Check Topology - Topology Rules
- Check Unique ID - Format and Frequency
- **Address Point Layer Checks**
- Check MSAGComm Spaces
- Check RCLMatch
- Check Frequency
- Check ESN and Muni Attribute
- Check Next-Gen Against Legacy Fields
- **Road Centerline Layer Checks**
- Check MSAGComm Spaces
- Check Frequency
- Check Cutbacks
- Check Directionality
- Check Frequency (of dual carriageways)
- Find Overlaps
- Check Parities
- Check Next-Gen Against Legacy Fields

Expanded Validation Check Methodology

Check Feature Locations

- Function begins by establishing the outer boundary that will contain the features from the feature classes. For this function, the existence of the *COUNTY_BOUNDARY* feature class will supersede the *DISCREPANCYAGENCY_BOUNDARY* as the outer boundary. First, the function checks for the *DISCREPANCYAGENCY_BOUNDARY*. If the *COUNTY_BOUNDARY* does not exist, then the function will check for the `STATE` field within the *DISCREPANCYAGENCY_BOUNDARY* feature class. If the field does not exist, the field will be created and calculated as "OK". A feature layer is then created out of the resulting *DISCREPANCYAGENCY_BOUNDARY* feature class and dissolved in memory using the `STATE` field.

- Once the outer boundary has been established, the function will then loop through the available feature classes to query them by making feature layers. The query is `SUBMIT = 'Y'` for all feature classes. The *ROAD_CENTERLINE* feature class has an additional query to exclude features with topology exceptions. The *ADDRESS_POINT* feature class utilizes the selection method "COMPLETELY_WITHIN", while all other feature classes utilize the selection method "WITHIN". Once the feature classes have been queried, a SelectLayerByLocation ({selection method}) and a SelectLayerByAttribute ("SWITCH_SELECTION") are applied to the feature class with respect to the outer boundary to select all objects outside the outer boundary. If the count for the selection is greater than zero, an error is reported for those objects that are selected that have non-null unique IDs.

Check Topology

- First, the function checks for and deletes pre-existing Topology and Topology error feature classes (i.e., errors that produce Points, Lines, and Polygons). The function then recreates the Topology and exports the errors to feature classes within the GDB.

- Next, the function loops through the Topology error feature classes. A feature layer is then created from the current error feature class (polygon, line, or point) to query to where `isException = 0`. If an error is returned (i.e., the fast count of the feature layer is greater than zero), the function searches through the feature layer with the error fields. The function assigns the appropriate parameters to default script parameters. General, most topology rules use only the Origin or Destination (i.e., there is only one feature class associated with the error type). The standard parameter setup includes:

    - `origin_fc` and `destination_fc` (feature class name describing a given Origin or Destination), `origin_id` and `destination_id` (`OBJECTID` for the object associated with the error within the feature class), `origin_id_field_name` and `destination_id_field_name` (error feature class origin and destination `OBJECTID` field name), and `ruleDesc` (description of the topology rule to error originated from).

- The function then checks to see if we have already looked at a given (`rule`, `origin_fc`, `destination_fc`). If not, then the tuple is added to the already_checked list.

- Next, the function then checks to make sure `origin_fc` and `destination_fc` are not None or an empty string or a blank space.
    - If both fc fields exist:
        - First, the function then creates a feature layer from current error feature layer with the query of `RuleDescription = '{current rule}'`. Then, the function creates table views from the `origin_fc` and `destination_fc` that will have the `OBJECTID` calculated to a `RecordID` field for the purposes of joining both feature classes to the error feature layer. The function then adds the two joins to the error feature layer. A query is then concatenated and field list created to accumulate a list of (`rule`, `origin_fc`, `origin_unique_id`, `destination_fc`, `destination_unique_id`).
    - If only one fc field exists:
        - First, the function then creates a feature layer from current error feature layer with the query of `RuleDescription = '{current rule}'`. Then, the function creates table views from the `fc`. The function then joins the FL to the error feature layer. A query is then concatenated and field list created to accumulate a list of (`rule`, `fc`, `unique_id`, "", "").
- The function then looks at the accumulated list of tuples returned from all error feature layers to count and concatenate the appropriate error messages to pass to the *RecordResults* function.

Check RCLMatch

- First, it should be noted that the `MSAGComm` fields are used for the `RCLMatch` comparison between *ADDRESS_POINT* and *ROAD_CENTERLINE* feature classes. The function for the check first creates the list of fields that will be used for the `RCLMatch` comparison including the `NAME_COMPARE` field. These are the same fields used in the `FullName`/`FullAddr` concatenation, so there shouldn't be any issues.
- A table view is created out of the *ROAD_CENTERLINE* feature class to query down to `SUBMIT = 'Y'` and then a *rcTable* table is created from the table view to be used in the `prep_roads_for_comparison` function. `Prep_roads_for_comparison` is then run for the *rcTable* table with the appropriately specified parameters. The process is then repeated for the *ADDRESS_POINT* feature class with a query of `SUBMIT = 'Y' AND RCLMatch <> 'NO_MATCH'` and a table named *apTable*.
- `Error: RCLSide is null`: The analysis begins with the selection of objects in the *apTable* where `RCLSide IS NULL`. If `RCLSide` is null, the unique ID for the object is returned to the no_rclside error list. The selection is then cleared for further analysis.
- `Error: RCLMatch is reporting a NENA Unique ID that does not exist in the road centerline`: A join is completed between the *apTable* `RCLMatch` field and the *rcTable* unique ID field. A SelectLayerByAttribute is performed where the `rcTable.NGUID_RDCL IS NULL`. If the unique ID field is null, the `RCLMatch` value did not have a matching unique ID in the *rcTable* and so

the *apTable* unique ID for the object is return to the ngsegid_doesnt_exist error list. The selection is then cleared for further analysis.

- `Error: RCLMatch does not correspond to a NENA Unique ID that matches attributes`: This analysis looks at both the left and right sides of the road centerline to see if the CODE_COMPARE fields resulting from the `prep_roads_for_comparison` function for *rcTable* and *apTable* match. A SelectLayerByAttribute is performed on the joined table to query down to `RCLSide = {current side} AND rcTable.NGUID_RDCL is not null AND apTable.CODE_COMPARE <> rcTable.CODE_COMPARE_{current side}`. If objects are selected, the *apTable* unique ID is added to the streets_or_msags_dont_match error list. The selection is then cleared for further analysis.

- `Error: Address does not fit in range of corresponding RCLMatch/Error: Road segment address ranges include one or more null values`: For a SelectLayerByAttribute on the joined table with the query of `RCLSide = {current side} AND rcTable.NGUID_RDCL is not null AND apTable.CODE_COMPARE = rcTable.CODE_COMPARE_{current side}`, this analysis compares the `Address` field in *apTable* to the `From/To` and `Parity` values in *rcTable* for a given road side. First, the function checks to see if the `From/To` values are not None type. If the `From/To` values are null, the *apTable* unique ID is added to the null_values error list. Next, the `Parity` value determines the range_counter (1 for "B", 2 for "O"/"E") to create a list of possible values using `From/To` values for a given side of the road segment. If the `Address` values is not within the range list, the *apTable* unique ID is added to the doesnt_match_range error list.

- The error lists of unique IDs are then parsed to the appropriate error string and returned to the *FieldValuesCheckResults* table.

Check Frequency

- First, depending on whether we are looking at *ADDRESS_POINT* or *ROAD_CENTERLINE*, a table view will be made using the appropriate query (AP: `Address <> 0 AND SUBMIT = 'Y'`; RDCL: `Add_L_From <> 0 AND Add_L_To <> 0 AND Add_R_From <> 0 AND Add_R_To <> 0 AND SUBMIT = 'Y'`). A field concatenation is then created for the Statistics_analysis to get the count for the concatenation of the selected fields. A table view is then created where `FREQUENCY > 1`. If there are records returned in the table, then a duplicate concatenation exists and analysis to determine the unique IDs is performed. A full frequency check will return Errors, while a partial (dual carriageway) frequency check will return Notices.

Check ESN and Muni Attributes

- First, the function checks for the existence of the *ADDRESS_POINT* feature class and creates a feature layer with the query `SUBMIT = 'Y'`. Then, the function checks for the ESZ (required) and Municipality (optional) feature classes inside the NG911 dataset. If the feature class exists, the function searches through the objects within the feature classes with the query `SUBMIT = 'Y'`. A query is then concatenated to make a feature layer of the current object using the unique ID. Then,

a SelectLayerByLocation is performed between the address point feature layer and the current selected object. The function then searches through the selected address points. Next, the unique ID for the selected address point needs to be non-null. If this is the case, the function checks if the value of specified field for the current address point object is not equal to the value of the selected `ESN` or `City` field in the current feature layer. Notices are then returned when appropriate.

Find Overlaps

- First, the function creates, calculates, and trims a string for the `NAME_OVERLAP` field in the *ROAD_CENTERLINE* feature class that will be used in the check analysis. The fields used in the concatenation are the same are as the ones used in the creation of the `FullName` field. The function then searches through the *ROAD_CENTERLINE* feature class with a query `SUBMIT = 'Y'`. The function checks to make sure the `MSAGComm` fields are not null or empty strings or a blank space. If the current segment of the road and side has not already been checked, the function then utilizes a MSAG label check function called `checkMsagLabelCombo`.
- The `checkMsagLabelCombo` function searches through the road centerline features that match the query created by concatenating a string using the values passed to the function. The function then determines the range for the those features that match the query. If the value in the range is not in the already-looked-at values for the segment and side, then the value is added to the already-looked-at list. Otherwise, the unique ID is then added to the overlaps error list. The overlap errors list is then returned to the main `FindOverlaps` function.
- If there are overlaps returned, the main `FindOverlaps` function will then make a feature layer of the *ROAD_CENTERLINE* feature class using the return unique IDs for the overlaps to create the query. An notice is then created for all the returned objects.

Usage

1. Open ArcCatalog and navigate to the toolbox called “Oklahoma NG911 GIS Tools”, expand the toolbox, then expand the toolset called “Validation Tools.” Use the tools in the numerical order presented with the following guidelines.
2. In the “Geodatabase” parameter, select the geodatabase of data to be checked.
3. Check which data checks you want to run. When running each tool for the first time, we recommend choosing all options.
4. Run the tool.
5. Alternatively, to run all checks, open and run 8 Check All Required.
6. The basic results of the data checks are shared in the ArcGIS dialog box. A simple Pass/Fail text file is created in the folder containing the geodatabase called "[Geodatabase_Name].txt" when CheckAll validations are run. The detailed results of the data checks will appear in two tables that are added to your geodatabase: *TemplateCheckResults* & *FieldValuesCheckResults*. The results reported in these tables will accumulate until you run the script titled 6 Optional Clear Results Table.
7. Based on the results of the data check, you can edit your data as necessary.

8. After data is edited, the necessary data checks can be rerun.
9. The script called 7 Optional Verify Topology Exceptions will double-check that all road centerline topology errors are recorded as exceptions in the data and the topology.
10. The script called 7.1 ESB Gap Locations will look for the areas where the ESB Boundary layers are not overlapping with the Discrepancy Agency Boundary.

Support Contact

For issues or questions, please contact through email Riley Baird at rbaird@odot.org or Emma Baker at ebaker@odot.org with the Oklahoma Transportation Cabinet, and please include in the email which script you were running, any error messages, and a zipped copy of your geodatabase. Change the file extension from `zip` to `piz` so it gets through the email server. If there are further data transfer issues, contact Emma or Riley to make alternative data transfer arrangements.

Disclaimer

MSAG Toolset

Toolset Credits

- Riley Baird, Oklahoma Transportation Cabinet
- Emma Baker, Oklahoma Transportation Cabinet

Last Revised:

March 17, 2025

Abstract

The MSAG Toolset compares MSAG/Telephone data to NG911 feature class data.

Tools

MSAG/TN-NG911 Comparison

- Compares MSAG and/or Telephone file(s) with the NG911 Standard feature class(es), Road Centerline and/or Address Point, respectively. Utilizes excel or csv files.

Usage

1. Select Standard geodatabase.
2. Select analysis.
3. Select required and/or optional parameters.
4. Execute the tool.

Support Contact

For issues or questions, please contact through email Riley Baird at rbaird@odot.org or Emma Baker at ebaker@odot.org with the Oklahoma Transportation Cabinet, and please include in the email which script you were running, any error messages, and a zipped copy of your geodatabase. Change the file extension from `zip` to `piz` so it gets through the email server. If there are further data transfer issues, contact Emma or Riley to make alternative data transfer arrangements.

Disclaimer

The Oklahoma NG9-1-1 GIS Toolbox is provided by the Oklahoma Geographic Information (GI) Council, Oklahoma 9-1-1 Management Authority, Oklahoma Department of Transportation (ODOT), Oklahoma Office of Geographic Information (OGI) , and associated contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the Oklahoma GI Council, Oklahoma 9-1-1 Management Authority, ODOT, OGI, or associated contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

Comparison Toolset

Toolset Credits

- Riley Baird, Oklahoma Transportation Cabinet
- Emma Baker, Oklahoma Transportation Cabinet

Last Revised:

March 17, 2025

Abstract

The Comparison Toolset can be used to look for differences between versions of NG911 data feature classes or whole geodatabases.

Tools

Compare NG911 Feature Classes

- Compares feature classes.

Usage

1. Select Standard geodatabase.
2. Select required and/or optional parameters.
3. Execute the tool.

Compare NG911 Geodatabases

- Compares user-specified geodatabases.

Usage

1. Select Standard geodatabase.
2. Select required and/or optional parameters.
3. Execute the tool.

Support Contact

For issues or questions, please contact through email Riley Baird at rbaird@odot.org or Emma Baker at ebaker@odot.org with the Oklahoma Transportation Cabinet, and please include in the email which script you were running, any error messages, and a zipped copy of your geodatabase. Change the file extension from `zip` to `piz` so it gets through the email server. If there are further data transfer issues, contact Emma or Riley to make alternative data transfer arrangements.

Disclaimer

The Oklahoma NG9-1-1 GIS Toolbox is provided by the Oklahoma Geographic Information (GI) Council, Oklahoma 9-1-1 Management Authority, Oklahoma Department of Transportation (ODOT), Oklahoma Office of Geographic Information (OGI) , and associated contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the Oklahoma GI Council, Oklahoma 9-1-1 Management Authority, ODOT, OGI, or associated contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

Submission Toolset

Toolset Credits

- Riley Baird, Oklahoma Transportation Cabinet
- Emma Baker, Oklahoma Transportation Cabinet

Last Revised:

March 17, 2025

Abstract

The data submission tools perform a variety of validation checks against the NG911 Data Model template to determine if the data is ready for submission. Notice that the geodatabase is unready for submission will be given if any validation checks are failed.

Tools

Check All and Zip

- Runs all required validation checks on user-specified geodatabase, and, if all checks are passed, the geodatabase is exported as a `.zip` file.

Usage

1. Select Standard geodatabase.
2. Select required and/or optional parameters.
3. Execute the tool.

Export to Shapefiles

- Exports all feature classes in user-specified geodatabase's `NG911` feature dataset as shapefiles so that they can be distributed independently of a geodatabase.

Usage

1. Select Standard geodatabase.
2. Select required and/or optional parameters.

3. Execute the tool.

Export to Zip

- Exports user-specified geodatabase as a `.zip` file.

Usage

1. Select Standard geodatabase.
2. Select required and/or optional parameters.
3. Execute the tool.

Support Contact

For issues or questions, please contact through email Riley Baird at rbaird@odot.org or Emma Baker at ebaker@odot.org with the Oklahoma Transportation Cabinet, and please include in the email which script you were running, any error messages, and a zipped copy of your geodatabase. Change the file extension from `zip` to `piz` so it gets through the email server. If there are further data transfer issues, contact Emma or Riley to make alternative data transfer arrangements.

Disclaimer

The Oklahoma NG9-1-1 GIS Toolbox is provided by the Oklahoma Geographic Information (GI) Council, Oklahoma 9-1-1 Management Authority, Oklahoma Department of Transportation (ODOT), Oklahoma Office of Geographic Information (OGI) , and associated contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the Oklahoma GI Council, Oklahoma 9-1-1 Management Authority, ODOT, OGI, or associated contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

Toolset Use Examples

Toolset Credits

- Riley Baird, Oklahoma Transportation Cabinet
- Emma Baker, Oklahoma Transportation Cabinet

Last Revised:

January 09, 2025

Abstract

The possible uses for the Toolset vary based on the current configuration of your data.

Dataset is completely standards-compliant

- In this scenario, the user would first run the Validate All Tool.

Validate All Tool

- After this, depending on the results, the user may need to alter the data in order to match Oklahoma NG911 Standards. There are some tools that can assist in this process (Enhancement).

Dataset is partially standards-compliant

- In this scenario, the user may want to run the Validate All tool to find out what in the dataset is not standards-compliant. Depending on the results, the user could either begin with the OK Prep tool "0-Create GDB" or the appropriate Field Map tool.
- The Create GDB tool creates a new standards-compliant GDB and the user can selected the appropriate standards-compliant feature classes. If the user would like to create blank standards-compliant feature classes, those options are available. Note: if the submitted feature class is not standards-compliant, a blank feature class is created if the user has selected the appropriate option, otherwise the feature class is not created.

Create GDB Tool

- The Field Map tools create a standards-compliant feature class with the correct field parameters in a user-supplied output GDB and user-supplied field map. The user would then run the Validate Geodatabase tool again to figure out if the dataset needs further maintenance. There are some

tools that can assist in this process (Enhancement).

Field Map

Dataset is not standards-compliant

- In this scenario, the user would want to create a new standards-compliant GDB with no provided feature classes and the create blank feature class option turned **off**.
- The user would then use the Field Map tools to create the standards-compliant feature classes in the new standards-compliant GDB.
- The user would then run the Validate Geodatabase tool again to figure out if the dataset needs further maintenance. There are some tools that can assist in this process (Enhancement).

No existing dataset

- In this scenario, the user would want to create a new standards-compliant GDB with the create blank feature class option turned **on**.
- From there, the user would entered the appropriate data into the datasets as needed.

Support Contact

For issues or questions, please contact through email Riley Baird at rbaird@odot.org or Emma Baker at ebaker@odot.org with the Oklahoma Transportation Cabinet, and please include in the email which script you were running, any error messages, and a zipped copy of your geodatabase. Change the file extension from `zip` to `piz` so it gets through the email server. If there are further data transfer issues, contact Emma or Riley to make alternative data transfer arrangements.

Disclaimer

The Oklahoma NG9-1-1 GIS Toolbox is provided by the Oklahoma Geographic Information (GI) Council, Oklahoma 9-1-1 Management Authority, Oklahoma Department of Transportation (ODOT), Oklahoma Office of Geographic Information (OGI) , and associated contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the Oklahoma GI Council, Oklahoma 9-1-1 Management Authority, ODOT, OGI, or associated contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

NG911 Topology Rules

Last Revised:

January 09, 2025

Abstract

The Oklahoma NG911 Standards require that a number of topological relationships exist within and among feature classes. This document discusses these rules, the validation of which is implemented in the Toolkit's *Check Topology* functionality.

An explanation of the different topology rules can be found here.

Polygon Layer Rules

All polygon feature classes must individually conform to the rule:

- Must Not Overlap (Area)

ESB and PSAP Layer Rules

The ESB_EMS_BOUNDARY, ESB_FIRE_BOUNDARY, ESB_LAW_BOUNDARY, and PSAP_BOUNDARY must individually conform to the rule:

- Must Not Have Gaps (Area)

ROAD_CENTERLINE Layer Rules

The ROAD_CENTERLINE layer must conform to the following rules:

- Must Not Overlap (Line)
- *Must Not Have Dangles (Line)*
- Must Not Self-Overlap (Line)
- Must Not Self-Intersect (Line)
- Must Be Single Part (Line)

NOTE: The rule(s) in italics may be marked as exceptions on a per-feature basis.

Rules Involving the DISCREPANCYAGENCY_BOUNDARY Layer

Layer	Relationship to DISCREPANCYAGENCY_BOUNDARY
ADDRESS_POINT	Must Be Properly Inside (Point-Area)
ROAD_CENTERLINE	*Must Be Inside (Line-Area)*
ESB_EMS_BOUNDARY	Must Cover Each Other (Area-Area)
ESB_FIRE_BOUNDARY	Must Cover Each Other (Area-Area)
ESB_LAW_BOUNDARY	Must Cover Each Other (Area-Area)
ESZ_BOUNDARY	Must Cover Each Other (Area-Area)
PSAP_BOUNDARY	Must Cover Each Other (Area-Area)

NOTE: The rule(s) in italics may be marked as exceptions on a per-feature basis.

Exceptions

The ROAD_CENTERLINE feature class includes the `TopoExcept` field, which allows the user to mark individual features as exempt from the *Must Not Have Dangles (Line)* and/or the *Must Be Inside (Line-Area)* (with DISCREPANCYAGENCY_BOUNDARY) rule. These exceptions will be accounted for when the Verify Topology Exceptions Validation tool is run.

Support Contact

For issues or questions, please contact through email Riley Baird at rbaird@odot.org or Emma Baker at ebaker@odot.org with the Oklahoma Transportation Cabinet, and please include in the email which script you were running, any error messages, and a zipped copy of your geodatabase. Change the file extension from `zip` to `piz` so it gets through the email server. If there are further data transfer issues, contact Emma or Riley to make alternative data transfer arrangements.

Disclaimer

Oklahoma Next-Generation 9-1-1 Pro Toolkit

Credits

Oklahoma Adaptation

Scripts written by Riley Baird and Emma Baker with the Oklahoma Transportation Cabinet (OTC)

Last Revised:

August 18, 2025

Abstract

This is a log of changes to the NG911 GIS Pro Toolkit since Version 3.0.0.

Most Recent Changes

Version 3.0.0

- First general availability release!
- Issue #17 - Fixed problem with RCLMatch/RCLSide analysis.

Previous Changes

Version 3.0.0-beta.1

- Issue #2 - False uniqueness errors for certain cases involving road centerlines fixed.
- Issue #17 - **Still outstanding** as of this release; partially addressed. See this comment on #17 for workaround.
- Improvements and error-code changes to *Check ESN Attribute (Road Centerline)* validation routine.

Version 3.0.0-alpha.5

- Issues #5 & #16 - Reworked the *Check ESN Attribute (Road Centerline)* validation routine's logic
- Issue #9 - Removed unnecessary *Check MSAGComm Consistency* validation routines
- Issue #10 - Reworked the *Check Address Range Directionality* validation routine's logic
- Issue #16 - Added `fill_value` of `"0"` for fields with roles `esn`, `esn_l`, and `esn_r` to `config.yml`

- Issue #18 - Removed non-functional fishbone analysis tool
- Issue #19 - Most tools now have output parameters
- Issue #21 - Improved NGUID format diagnostic messages
- Reworked the *Check Topology* validation routine's logic
- Improved *Validate Geodatabase* interface

Version 3.0.0-alpha.4

- Issue #11 - Rebuilt `4 - Create Standard ESZ & ESB Feature Classes`, fixing a crash due to a field-type/value-type mismatch.
- Issue #13 - Fixed an issue in `Validate Geodatabase` causing false-positive reports of `ERROR:GENERAL:UNIQUENESS` with address points.
- Issue #20 - Fixed an issue where some fields were not being copied over during field mapping.

Version 3.0.0-alpha.3

This update included:

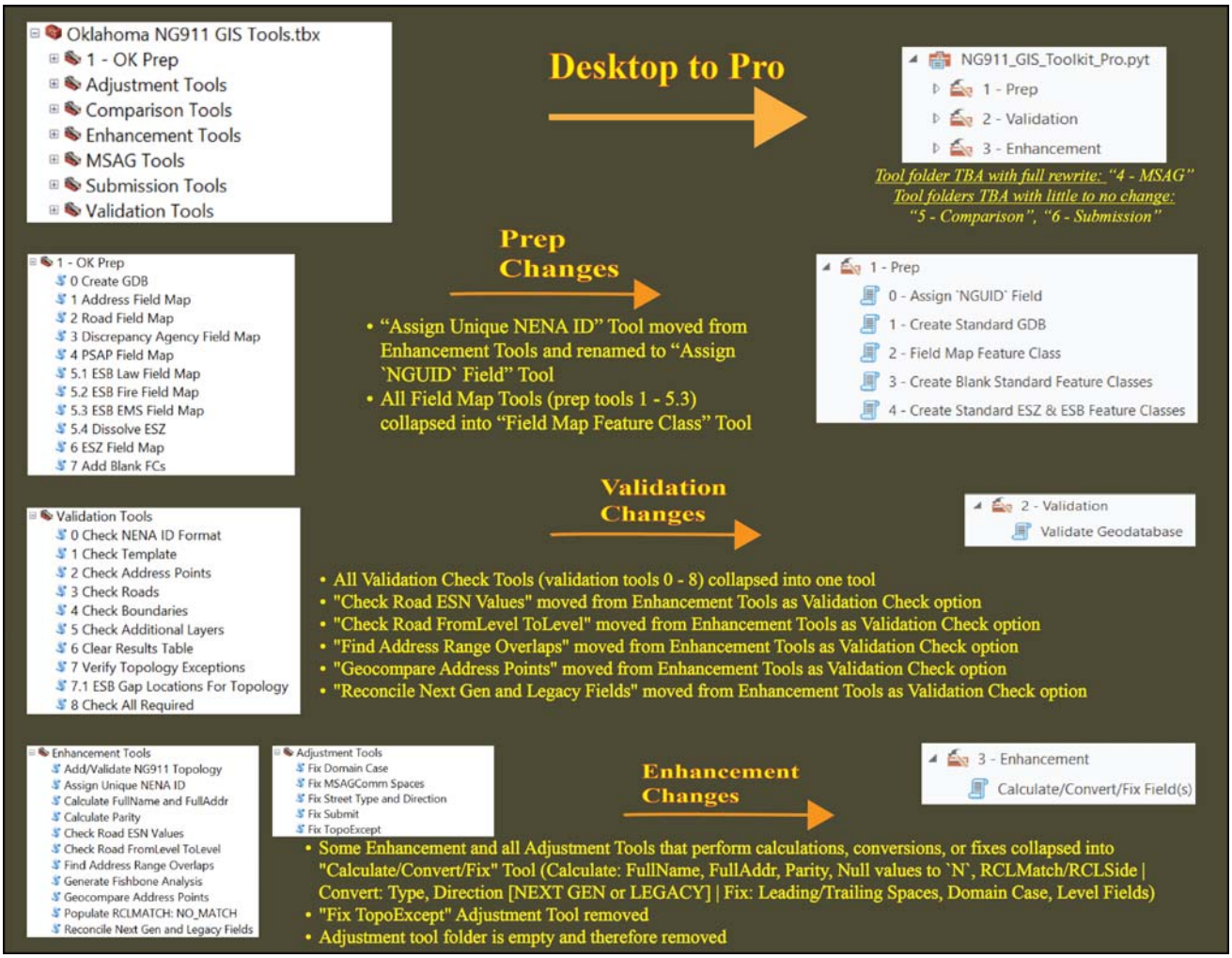
- Validation routine *Check Topology* was added to `Validate Geodatabase`.

Version 3.0.0-alpha.2

This update included:

- MSAG (4) tool has been written - `MSAG/TN-NG911 Comparison`.
- Comparison (5) tools have been written - `Compare NG911 Feature Classes`, `Compare NG911 Geodatabases.`
- Submission (6) tools have been written - `Check All and Zip`, `Export to Shapefiles`, `Export to Zip`.
- Validation tool output fixed.
- Enhancement Tool - `Add Topology` has been written.
- Enhancement Tool - `Generate Fishbone Analysis` has **not** been written.
- Documentation updated but not completely.

Version 3.0.0-alpha.1



This update included:

- Convert from **Desktop** version of the Toolkit to **Pro** version of the Toolkit.
- Collapse all Adjustment and most Enhancment tools into one Enhancement tool.
- Validation tools collapsed into one tool.
- Assign NGUID tool moved to Prep tools.
- All Field Map tools collapsed into one tool.
- Comparison, MSAG, and Submission tools have not been written.
- Generate Fishbone Analysis tool has not been written.
- Toolkit open for alpha test.

Support Contact

For issues or questions, please contact through email Riley Baird at rbaird@odot.org or Emma Baker at ebaker@odot.org with the Oklahoma Transportation Cabinet, and please include in the email which script you were running, any error messages, and a zipped copy of your geodatabase. Change the file extension from `zip` to `piz` so it gets through the email server. If there are further data transfer issues, contact Emma or Riley to make alternative data transfer arrangements.

Disclaimer

The Oklahoma NG9-1-1 GIS Toolbox is provided by the Oklahoma Geographic Information (GI) Council, Oklahoma 9-1-1 Management Authority, Oklahoma Department of Transportation (ODOT), Oklahoma Office of Geographic Information (OGI) , and associated contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the Oklahoma GI Council, Oklahoma 9-1-1 Management Authority, ODOT, OGI, or associated contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

Error Glossary

Toolset Credits

- Riley Baird, Oklahoma Transportation Cabinet
- Emma Baker, Oklahoma Transportation Cabinet

Last Revised:

January 09, 2025

Abstract

This document lists and describes the various messages written to the geoprocessing console, the *TemplateCheckResults* table, and the *FieldValuesCheckResults* table. In cases where the exact text of a message may vary, the contents of the variable part(s) are described in `{curly braces}`.

Tool Failure Errors are errors that print to the groprocessing console when a tool fails.

Validation Errors describe issues that impede submission. **Validation Notices** do not impede submission.

Tool Failure Errors

- `RuntimeError: [ArcGIS License Error] {message}` - The NG911 tool uses an ArcGIS geoprocessing tool that requires a higher level license. See the documentation for the NG911 tool to see if it can be run with different options that require a lower license level. Otherwise, **there is no workaround**; a higher ArcGIS license is required.
- `ExecuteError: {message}` - Generated by ArcGIS; may be accompanied by a six-digit error code. These errors may occur if the NG911 tool is being used improperly; refer to the NG911 tool's documentation. If such an error occurs despite proper user of the tool, please report a bug (see Support Contact below).
- `{error type}: BUG ENCOUNTERED. REPORT TO DEVELOPERS {message continues...}` - The script encountered a situation internally that should not have happened. Please report a bug (see Support Contact below) and **include all output in the geoprocessing console** in the report.

Template Validations

Check Layer List Ignores `SUBMIT`

- `Error: No feature dataset named 'NG911' exists` - The NG911 geodatabase does not contain a feature dataset named *NG911*. (*TemplateCheckResults*)
- `Error: Required layer {feature class name} is not in geodatabase dataset.` - A required feature class with the indicated name does not exist in the *NG911* feature dataset. (*TemplateCheckResults*)

If this check generates any errors, a console warning is printed: `Not all required geodatabase datasets and/or layers are not present. See TemplateCheckResults.`

Check GDB Domains Ignores `SUBMIT`

- `Error: Domain Name {domain name} is not an approved domain.` - The geodatabase contains a domain with a non-standard name. (*FieldValuesCheckResults*)
- `Error: Domain Coded Value '{code}' for {domain name} Domain is incorrect.` - A coded value that is not Standards-compliant exists in a domain. (*FieldValuesCheckResults*)
- `Error: Domain Coded Value Description '{existing code description}' for {code} coded value in {domain name} Domain is incorrect and should be '{standard code description}'.` - The description for a coded value is not Standards-compliant. The descriptions in the message may be abbreviated. (*FieldValuesCheckResults*)
- `Notice: Domain Description '{existing domain description}' for {domain name} Domain should be '{standard domain description}'.` - The domain description for a Standards-compliant domain name is not Standards-compliant. The descriptions in the message may be abbreviated. (*FieldValuesCheckResults*)

If this check generates any errors, a console warning is printed: `Completed checking GDB domains: {#} issues found. See table FieldValuesCheckResults for results.`

Check Spatial Reference Ignores `SUBMIT`

- `Error: Spatial reference of feature dataset is incorrect.` - The *NG911* feature dataset does not have the correct spatial reference. (*TemplateCheckResults*)

If this check generates any errors, the above error is also printed to the console as a warning.

Check Required Fields Ignores `SUBMIT`

- `Error: {feature class name} does not have required field {field name}` - A field required by the Standards does not exist in a feature class. (*TemplateCheckResults*)
- `Error: HNO/Address field of Address Points is not an integer or a double, it is a {field data type}` - The data type of the `Address` field in the *ADDRESS_POINT* feature class

should be numeric. (*TemplateCheckResults*)

If a required feature class does not exist inside the *NG911* feature dataset, a console warning is printed: `Required layer {feature class name} does not exist`

If this check generates any errors, a console warning is printed: `Completed check for required fields: {#} issues found. See table FieldValuesCheckResults for results.`

Check Required Field Values Uses SUBMIT

- `Error: {field name} is null for Feature ID {NGUID}` - A required field is null for a specific feature. (*FieldValuesCheckResults*)

If a feature class does not have a field named `SUBMIT`, a console warning is printed: `Cannot check required field values for {feature class name}`

If a feature class is missing one or more required fields, a console warning is printed: `Could not check all fields in {feature class name}. Looking for: {list of required fields}`

If a feature class has no features where `SUBMIT` is `Y`, a console warning is printed: `{Feature class name} has no records marked for submission. Data will not be verified.`

If a required feature class does not exist inside the *NG911* feature dataset, a console warning is printed: `Required layer {feature class name} does not exist`

If this check generates any errors, a console warning is printed: `Completed check for required field values: {#} issues found. See table FieldValuesCheckResults for results.`

Check Submission Numbers Uses SUBMIT

- `Error: {feature class name} has 0 records for submission` - A required feature class has no features where `SUBMIT` is `Y`. (*TemplateCheckResults*)

If a feature class does not have a field named `SUBMIT`, a console warning is printed: `SUBMIT field does not exist in required layer {feature class name}`

If a required feature class does not exist inside the *NG911* feature dataset, a console warning is printed: `Required layer {feature class name} does not exist`

If this check generates any errors, a console warning is printed: `One or more layers had no features to submit. See table TemplateCheckResults.`

Find Invalid Geometry Ignores SUBMIT

- `Error: Invalid geometry` - A feature does not contain the minimum necessary number of points per its indicated geometry. (Point: ≥1, Line: ≥2, Polygon: ≥3) (*FieldValuesCheckResults*)

If a feature class has no NGUID field, a console warning is printed: `NGUID field {field name} does not exist in {feature class name}`

If a required feature class does not exist inside the *NG911* feature dataset, a console warning is printed: `Required layer {feature class name} does not exist`

If this check generates any errors, a console warning is printed: `Completed for invalid geometry: {#} issues found. See FieldValuesCheckResults.`

Check Values Against Domain Uses SUBMIT

- `Error: Value {field value} not in approved domain for field {field name}` - A field contains a value (other than null or an empty string) that is not in the Standards-compliant domain. (*FieldValuesCheckResults*)
- `Error: Value {field value} not in approved domain for field {field name}` - The value of the `Address` field is either less than 0 or greater than 999999. (*FieldValuesCheckResults*)

If a feature class has no features where `SUBMIT` is `Y`, a console warning is printed: `No features are marked for submission in {feature class name}. Please mark records for submission by placing Y in the SUBMIT field.`

If a feature class does not have a field named `SUBMIT`, a console warning is printed: `Cannot check required field values for {feature class name} because the SUBMIT field does not exist.`

If a feature class is missing one or more fields with domains, a console warning is printed: `Field {field name} in feature class {feature class name} does not exist, and its values cannot be checked against domain {domain name}.`

If a required feature class does not exist inside the *NG911* feature dataset, a console warning is printed: `Required layer {feature class name} does not exist`

If this check generates any errors, a console warning is printed: `Completed checking fields against domains: {#} issues found. See table FieldValuesCheckResults for results.`

Check Feature Locations Uses SUBMIT

- `Error: Feature not inside discrepancy agency boundary` - A feature was found to not be within any *DISCREPANCYAGENCY_BOUNDARY* feature, **and** the feature does not have a `TopoExcept` value of `INSIDE_EXCEPTION` or `BOTH_EXCEPTION` (if applicable). Detailed information on topology rules involving the *DISCREPANCYAGENCY_BOUNDARY* feature class can be found in the NG911 Topology Rules documentation. (*FieldValuesCheckResults*)

If neither *DISCREPANCYAGENCY_BOUNDARY* (in the *NG911* feature dataset) nor *COUNTY_BOUNDARY* (in the *OptionalLayers* feature dataset) exist, a console warning is printed: `Check Feature Locations`

`could not run because the discrepancy agency and/or county boundary feature classes are absent or misnamed.`

If this check generates any errors, a console warning is printed: `{Feature class name}: issues with some feature locations`

If this check generates any errors, a console warning is printed: `Completed check on feature locations: {#} issues found. See table FieldValuesCheckResults.`

Check Topology Uses SUBMIT

- `Error: Both origin and destination feature class names are null.` - Both Origin and Destination feature class name fields in an error feature class are None, a blank string, or a space. (*FieldValuesCheckResults*)
- `Error: Topology issue- {topology rule} | Number of errors- {#}` - A topology rule was violated, and the violating feature was not marked as an exception with its `TopoExcept` field (if applicable). Detailed information on topology rules can be found in the NG911 Topology Rules documentation. (*FieldValuesCheckResults*)

If this check generates any errors, a console warning is printed: `Topology check complete. {#} issues found. Results in FieldValuesCheckResults.`

Check Unique ID Format and Frequency Ignores SUBMIT

- `Error: {NGUID} is a duplicate ID` - The same NGUID was found on multiple records (i.e. that unique ID is not unique). (*FieldValuesCheckResults*)
- `Error: Unique ID format wrong.` - The unique ID is not in the Standards-compliant format. (*FieldValuesCheckResults*)
- `Error: {#} records with null Unique IDs.` - One or more features have a null* unique ID attribute. (*FieldValuesCheckResults*)

If a required feature class is not found, a console warning is printed: `{Feature class name} does not exist`

If this check generates any errors, a console warning is printed: `There are {#} records in {feature class name} with null or incorrectly-formatted unique IDs.`

If this check generates any errors, a console warning is printed: `Checked unique ID frequency. There were {#} issues. Results are in table FieldValuesCheckResults.`

Address Point Validations

If the *ADDRESS_POINT* feature class was not found in its expected location, a console warning is printed:

`Layer ADDRESS_POINT does not exist and therefore cannot be checked. This will prevent submission.`

Check MSAGComm Spaces Uses SUBMIT

- `Error: {MSAGComm field name} has a leading or trailing space.` - A feature's `MSAGComm`, `MSAGComm_L`, or `MSAGComm_R` attribute begins or ends with a space. `MSAGComm`, `MSAGComm_L`, and `MSAGComm_R` attributes should consist neither of a space followed by one or more characters nor of one or more characters followed by a space. (*FieldValuesCheckResults*)
- `Notice: {MSAGComm field name} is a blank string or has only a space.` - A feature's `MSAGComm`, `MSAGComm_L`, or `MSAGComm_R` attribute is not null, but consists of a text string either with no characters or with a single space as the only character. (*FieldValuesCheckResults*)

If this check generates any errors, a console warning is printed: `Check complete. {#} issues found. See table FieldValuesCheckResults for results.`

Check RCLMatch Uses SUBMIT

- `Error: RCLMatch is reporting a NENA Unique ID that does not exist in the road centerline` - The `RCLMatch` attribute of an *ADDRESS_POINT* feature is not null, but there is no *ROAD_CENTERLINE* feature with that `NGUID_RDCL` attribute. (*FieldValuesCheckResults*)
- `Error: RCLMatch does not correspond to a NENA Unique ID that matches attributes` - The matching *ROAD_CENTERLINE* feature's street-name-related or `MSAGComm` attributes don't match those of the *ADDRESS_POINT* feature. (*FieldValuesCheckResults*)
- `Error: Address does not fit in range of corresponding RCLMatch` - The *ADDRESS_POINT* feature's `Address` attribute is not within the matching *ROAD_CENTERLINE* feature's address range (defined by `Add_L_From`, `Add_L_To`, `Add_R_From`, `Add_R_To`). (*FieldValuesCheckResults*)
- `Error: Road segment address ranges include one or more null values` - One or more of the matching *ROAD_CENTERLINE* feature's `Add_L_From`, `Add_L_To`, `Add_R_From`, and `Add_R_To` attributes are null instead of numeric. (*FieldValuesCheckResults*)
- `Error: RCLSide is null` - The *ADDRESS_POINT* feature's `RCLSide` attribute is null. (*FieldValuesCheckResults*)

If the *ADDRESS_POINT* feature class does not have an `RCLMatch` field, a console warning is printed: `Missing required field RCLMatch.`

If this check generates any errors, a console warning is printed: `Check complete. {#} issues found. See table FieldValuesCheckResults for results.`

Check Frequency Uses SUBMIT

- `Error: Frequency table exists; please delete or close and rerun the check.` - The frequency table exists, and the tool failed to delete it. (*FieldValuesCheckResults*)
- `Error: {NGUID} has duplicate field information` - A record has a combination of certain attributes identical to that of another record where this combination should be unique. For example, an error involving an *ADDRESS_POINT* feature means there are multiple features represent the same address. (*FieldValuesCheckResults*)
- `Error: Could not complete duplicate record check. {Error technical information}` - The program encountered an execution error. (*FieldValuesCheckResults*)
- `Notice: {NGUID_RDCL} has duplicate address range information` - Multiple *ROAD_CENTERLINE* features have one or more of their `Add_L_From`, `Add_L_To`, `Add_R_From`, and `Add_R_To` attributes that are identical. (*FieldValuesCheckResults*)

If the frequency table (*AP_Freq* or *Road_Freq*) exists and the script cannot delete it, a console warning is printed: `Please manually delete {frequency table name} and then run the frequency check again.`

If the `Address` field of the *ADDRESS_POINT* feature class is not of type `Integer` or `Double`, a console warning is printed: `Address field of Address Points is not an integer or a double field.`

If this check generates any errors, a console warning is printed: `Checked frequency. There were {#} duplicate records. Individual results are in table FieldValuesCheckResults`

Check ESN and Municipality Attribute Uses SUBMIT

- `Notice: Address point {OBJECTID} does not match {either "ESN" or "City"} in {either "ESZ_BOUNDARY" or "MUNICIPAL_BOUNDARY"} layer.` - (*FieldValuesCheckResults*)
- `Notice: Address point with OBJECTID {OBJECTID} does not have a NENA Unique ID. Its {either "ESN" or "City"} attribute was not checked against its containing {either "ESZ_BOUNDARY" or "MUNICIPAL_BOUNDARY"} polygon.` - (*FieldValuesCheckResults*)
- `Notice: ESN/Municipality check did not run. {Error technical information}` - The program encountered an execution error. (*FieldValuesCheckResults*)

If the *ADDRESS_POINT* feature class was not found in its expected location, a console warning is printed: `{Input path to ADDRESS_POINT} does not exist`

If the *ESZ_BOUNDARY* feature class was not found in its expected location, a console warning is printed: `ESZ layer does not exist. Cannot complete check.`

Check Next-Gen Against Legacy Fields Uses SUBMIT

- `Error: {Next-Gen field name} value {Next-Gen field value} does not match {Legacy field name} value {Legacy field value}.` - There is a mismatch between the

fully-written-out value in the Next-Gen field and the abbreviation in corresponding Legacy field. For example, a record with a `StreetType` attribute of `AVENUE` and a `LgcyType` attribute of `BLVD` would result in this error. (*FieldValuesCheckResults*)

If this check generates any errors, a console warning is printed: `Completed checking NG against Legacy field values: {#} issues found. See table FieldValuesCheckResults for results.`

Road Centerline Validations

Check MSAGComm Spaces Uses SUBMIT

See entry in Address Point Validations section.

Check Frequency Uses SUBMIT

See entry in Address Point Validations section.

Check Cutbacks Uses SUBMIT

- `Notice: This segment might contain a geometry cutback.` - A *ROAD_CENTERLINE* polyline feature has two adjacent segments that form an angle sharper than 55°. This **may** indicate a data error, or it may simply represent a road with an exceptionally sharp curve. (*FieldValuesCheckResults*)

If the *ROAD_CENTERLINE* feature class was not found in its expected location, a console warning is printed: `{Input path to ROAD_CENTERLINE} does not exist`

If this check generates any **notices**, a console warning is printed: `Completed check on cutbacks: {#} issues found. See FieldValuesCheckResults.`

Check Directionality Uses SUBMIT

- `Notice: Segment's address range is from high to low instead of low to high` - For the indicated feature, `Add_L_From` > `Add_L_To` and/or `Add_R_From` > `Add_R_To` instead of the other way around. (*FieldValuesCheckResults*)

If this check generates any **notices**, a console warning is printed: `Completed road directionality check. There were {#} issues. Results are in table FieldValuesCheckResults.`

Check Address Range Overlaps Uses SUBMIT

- `Notice: {MSAGComm field} needs to be a real value` - The value of `MSAGComm_L` or `MSAGComm_R` (as specified in the notice) is null*. (*FieldValuesCheckResults*)
- `Notice: {NGUID_RDCL} has an overlapping address range.` - A *ROAD_CENTERLINE* feature

overlaps an address range of another feature. (*FieldValuesCheckResults*)

If this check generates any **notices**, a console warning is printed: `{#} overlapping address range segments found. Please see {full path to AddressRange_Overlap output} for overlap results.`

Check Parities Uses `SUBMIT`

- `Error: Could not process parity check. Look for null values.` - One or more parity attributes are null. (*FieldValuesCheckResults*)
- `Error: Could not process parity check for a road segment with a null unique ID.` - A *ROAD_CENTERLINE* feature has a null unique ID, and a parity report could not be created. (*FieldValuesCheckResults*)
- `Error: One or more address ranges are null` - One or more address range attributes are null. (*FieldValuesCheckResults*)
- `Notice: {"L" or "R"} Side- Address range is 0-0, but the parity is recorded as {parity type} instead of Z` - The address range from and to attributes are both 0, so the parity should be set to `Z` (zero). (*FieldValuesCheckResults*)
- `Notice: {"L" or "R"} Side- Parity is Z (zero), but the address range is filled in with non-zero numbers.` - One or more address range attributes are non-zero, but the corresponding parity attribute is Z (zero). (*FieldValuesCheckResults*)
- `Notice: {"L" or "R"} Side- Parity is marked as {parity type} but the ranges filled in are {parity type} and {parity type}` - The parity attribute does not match the actual parities of the address range attributes. (*FieldValuesCheckResults*)
- `Notice: A wild error appeared! The wild error used {generic video game attack move}! It's a one-hit KO! {Message}` - The developers do not expect this situation to ever occur, but if it does, please email the support contacts listed below. (*FieldValuesCheckResults*)

If any *ROAD_CENTERLINE* feature's `Parity_L` or `Parity_R` attribute is null, a console warning is printed: `You have one or more parities set as null. Please populate those fields.`

If this check generates any errors, a console warning is printed: `Completed parity check. There were {#} issues. Results are in table FieldValuesCheckResults.`

Check Next-Gen Against Legacy Fields Ignores `SUBMIT`

- `Error: {Next-Gen field name} unabbreviated value {Next-Gen field value} does not match {Legacy field name} abbreviated value {Legacy field value}.` - There is a mismatch between the fully-written-out value in the Next-Gen field and the abbreviation in corresponding Legacy field. For example, a record with a `StreetType` attribute of `AVENUE` and a

LgcyType attribute of `BLVD` would result in this error. (*FieldValuesCheckResults*)

If this check generates any errors, a console warning is printed: `Completed checking NG against Legacy field values: {#} issues found. See table FieldValuesCheckResults for results.`

Note on Null

Where "null" is followed by an asterisk, "null" refers to any of the following: SQL `NULL`, Python `None`, a blank string (`''`), or a string consisting only of a single space (`' '`).

Support Contact

For issues or questions, please contact through email Riley Baird at rbaird@odot.org or Emma Baker at ebaker@odot.org with the Oklahoma Transportation Cabinet, and please include in the email which script you were running, any error messages, and a zipped copy of your geodatabase. Change the file extension from `zip` to `piz` so it gets through the email server. If there are further data transfer issues, contact Emma or Riley to make alternative data transfer arrangements.

Disclaimer