National Park Service
U.S. Department of the Interior

Natural Resource Stewardship and Science

NPS I&M Standard Operating Procedure: NPScape NPS Boundary-Derived Areas of Analysis SOP – Park, 3km, and 30km Areas of Analysis

Version #1.1

Cite As: Nelson, Lisa and Michelle Kinseth. 2015. NPScape NPS Boundary-Derived Areas of Analysis SOP – Park, 3km, and 30km Areas of Analysis. NPS/NRSS/IMD/SOP-1.1.

Summary

Using park boundaries derived through NRSS Bounds Processing (adapted from Lands Division), Areas of Analysis are created for use with NPScape Landscape Dynamics. These park-based Areas of Analysis are standard across the entire NPS, and exist for all IMD Parks as Park AOAs, 3km AOAs, and 30km AOAs.

Version History

Previous Version No.	Date of Revision	Changes	Made by	Reason	Updated Version No.
	20151014	Initial version	Michelle Kinseth		1.1

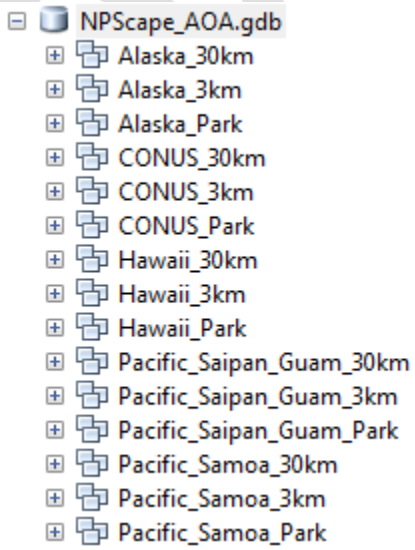
Contents

..... 1
Summary 1
Overview/Purpose 2
Requirements 3
Pre-Processing Requirements 3
AOA Processing – Standard NPScape AOAs 3
AOA Processing – Custom AOAs 5
Quality Control 5
Post-Processing 6
AOA Map Services – Creation and Publication 6
Literature Cited 7
Appendices 7
Appendix 1: Known issues 7
Appendix 2: Landscape Dynamics/NPScape Standard Areas of Analysis 7
Appendix 3: Scripts 7

Overview/Purpose

This NPScape SOP documents how **areas of analysis (AOAs)** are derived from NPS park boundaries for use in NPScape analyses. Three standard, park-based AOAs are used in the Landscape Dynamics (NPScape) Protocol: park AOAs, 3km AOAs, and 30km AOAs. The three and 30 kilometer AOA polygons are derived by buffering dissolved park polygons by three or thirty kilometers. This processing produces three AOAs for each NPScape park. An area attribute (AREA_SQKM) is calculated for each AOA polygon. See Appendix 3 for details.

A Python tool is used to create these AOAs as feature classes within feature datasets in a standardized file geodatabase: NPScape_AOA.gdb:



This AOA file geodatabase may be downloaded from the NPScape web site:
http://science.nature.nps.gov/im/monitor/npscape/methods_aoas.cfm

Note about Standard NPScape AOAs: Though these AOAs are derived from NPS Land Resources Division's Lands update, certain park AOA boundaries contained herein will differ from Lands based on local park/network staff knowledge. In these cases, the NPScape protocol defers to the local knowledge of park boundaries over the Lands version. As such, NPScape AOAs (specificallypPark AOAs) are ONLY valid for use with NPScape and the Landscape Dynamics Protocol (cite).

In some cases, Parks and/or Networks might have areas of interest for analysis that fall outside the standard NPScape AOAs (Park, 3km, 30km). Park and Network staff are provided the opportunity to provide these custom areas to NPScape for inclusion. As such, this SOP also provides instruction on creating Custom NPScape AOAs from park/network provided areas. These AOAs are organized in a separate file .gdb, by Network.

Requirements

Users of this SOP are required to have:

- minimum installation of ArcGIS 10.2.2
- Mapped drive, and read/write permissions *(inp2300fcsisro1\IMD_Data)*
- Database Connection with read permissions (Database Connections\NPScape_Stats.NPScape_Viewer.DBO_INPNISCVDBNRSST_IMDGIS.sde\NPScape_Stats.dbo.ParksNetworksRegions)
- Remote desktop permissions and access to GIS Server to publish Map Services (inp2300fcvhafo1)
- *Optional: IDE for Python (i.e. Pyscripter) with .pyt extension recognition – this is only necessary if editing the python toolbox; access limited to IMD Central Office GIS staff.*

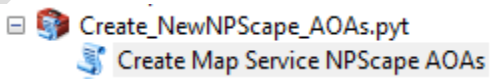
Pre-Processing Requirements

<u>Standard NPScape Bounds Processing:</u>
This tool requires the most recent version of *X:\ProjectData\Data_Processing\Bounds_Processing\Data\NPScape_Bounds.gdb\npscape_bounds_albers*, which requires NPScape Bounds Processing be completed first.
See NPScapeBounds_Processing_SOP.docx

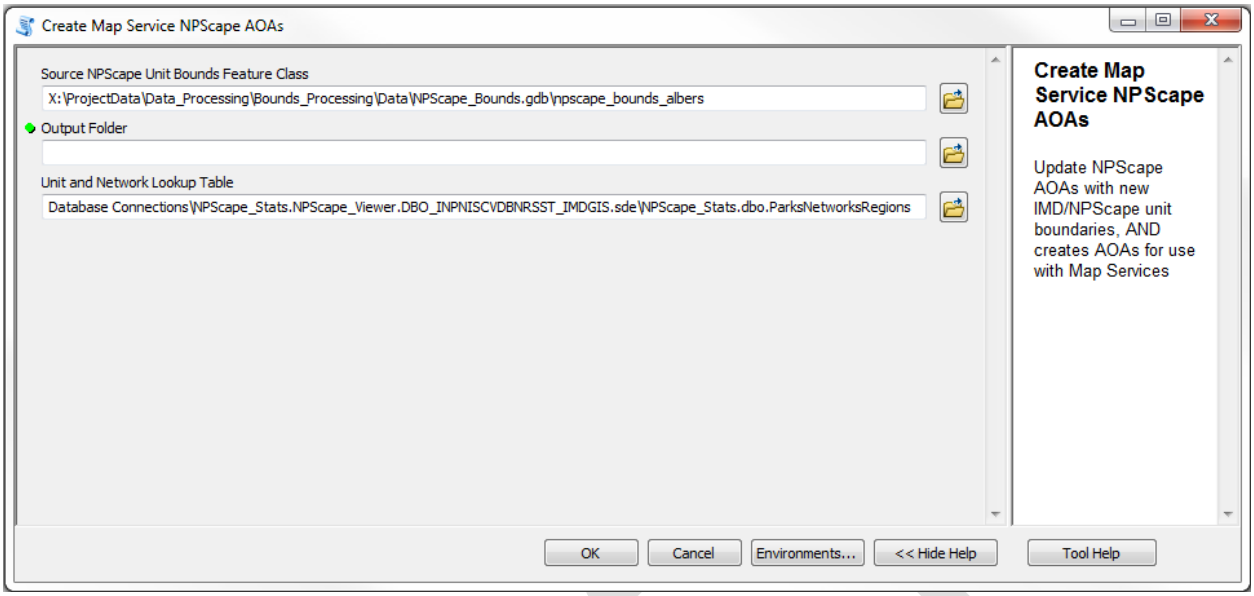
<u>Custom NPScape AOAs Processing:</u>
Requires Park/Networks to provide ahead of time any polygon(s) representing custom area(s) of interest for analysis. See SOP_CustomData_DRAFT.docx

AOA Processing – Standard NPScape AOAs

- To update standard NPScape AOA feature datasets and feature classes (Park, 3km, 30km) using the most recent NPScape Bounds features (derived from NPS boundary features through IMD Bounds Processing):
    - open ArcMap or ArcCatalog and run the tool **Create Map Service NPScape AOAs** (X:\ProjectData\Data_Processing\Bounds_Processing); the tool may take up to 2 hours to run.



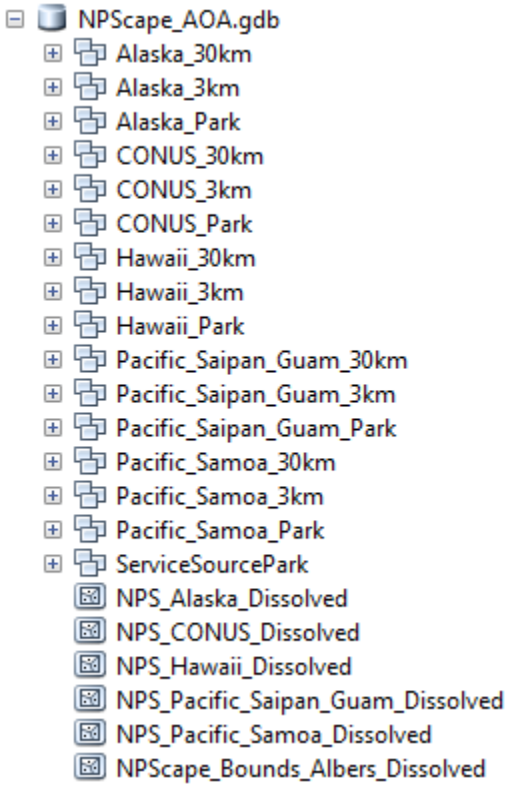
- Set Output Folder location to a local drive. **NOTE**: Make sure to run the tool locally. It will take 8+ hours if you attempt to create AOAs to a network drive.
- Select table/view "NPScape_Stats.dbo.ParksNetworksRegions" from NPScape_Viewer.Stats database (as noted in the figure below, see also Appendix 2).



<u>General processing steps of the tool **Create Map Service NPScape AOAs** (see Appendix 3 for script details):</u>
Create AOA file geodatabase, create standard AOA feature datasets (Park, 3km, 30km) with correct spatial reference (and transformation where applicable); dissolve source NPScape boundary feature class on UNIT_CODE; iterate through each dissolved park polygon, copying it to a new feature class in the relevant feature dataset, then buffering to 3 or 30km as appropriate; add AREA_SQKM attribute and calculate the area in square kilometers; import metadata; merge individual AOA polygons into a merged AOA feature class; finally, create dissolved feature classes of 3km and 30km features. The tool also creates Web Mercator versions of NPScape AOAs (Park, 3km, 30km), and copies these web mercator feature classes to GDB_NRSS.DBO.INPNISCVDBNRSST_IMDGIS.sde for use as map services in web map applications.

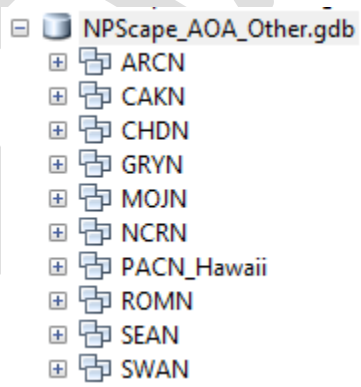
Standard AOA attribute table details for each feature class <*UNIT_CODE*>_AOA_<*aoa type*>, where <*aoa type*> = Park, 3km, or 30km:

UNIT_CODE = the park Unit Code
UnitName = the park name
NetworkCode = the Network Code home to the park
NetworkName = the full name of the Network home to the park
RegionName = the name of the Region home to the park
DATE_EDIT = date of last known edit of the park boundary
Feature_Source = author/creator of park boundary
Source_Metadata = link to data store (other) metadata for park boundary
BUFF_DIST (if applicable) = the buffered distance in meters (3000m or 30000m)
AREA_SQKM: total area of the AOA polygon in square kilometers
AOA_NAME (Map Service feature classes only) = <UNIT_CODE>_AOA_<aoa type> where aoa type = Park, 3km, or 30km



AOA Processing – Custom AOAs

To update and/or create new AOAs for custom, non-park-based areas of analysis submitted by Network staff, manually add custom AOAs to NPScape_AOA_Other.gdb. Custom AOAs are provided by park/network staff and should require minimum (to no) processing, as described in the Custom Data SOP. If processing is necessary: re-project AOA, Repair Geometry, add/edit any attribute fields, and import into the proper Network feature dataset (create as needed).



Quality Control

Using ArcMap, the AREA_SQKM values are spot-checked for a selected set of park-based AOAs. Using Field Calculator, the AOA area in square kilometers is calculated and checked against the AREA_SQKM value in the AOA feature class.

Using ArcMap, visually check the merged AOA feature class with the source NPScape Bounds polygons to check for spatial congruency. Check specific parks.

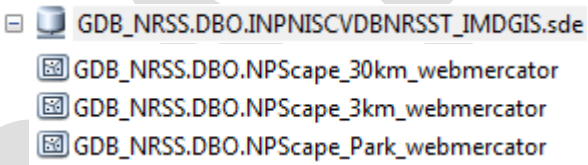
- Check for any known new or changed AOAs from last AOA update.
- Check Metadata on at least one feature class per feature dataset, edit as needed.

Post-Processing

- After QA/QC, COPY NPScape_AOA.gdb from local drive to X:\ProjectData\NPScape\DATA\SourceData\USData\Boundary AND X:\ProjectData\Data_Processing\Bounds_Processing\Data
- (Date stamp previous AOAs and save)? Or simply delete previous version from X drive?
- Version AOA geodatabase reference in Data Store and upload NEW .gdb
- Verify NPScape website link points to new version
- Rerun all NPScape Stats for new/updated/altered AOA boundaries. See NPScape Stats SOPs for each Landscape Dynamics Metric.

AOA Map Services – Creation and Publication

Standard AOAs Map Service - dissolved features for each (Park, 3km, 30km) are copied to SDE for use in a map service (GDB_NRSS.DBO.INPNISCVDBNRSST_IMDGIS.sde)



- Remote Desktop to inp2300fcvhafo1
  - Delete existing NPScapeAOA map service from NPScape folder
  - Open NPScape_MapService.mxd and confirm any new/changed AOAs are reflected (confirm the copy of web Mercator features to SDE worked)
    - Edit Data Frame Properties (Description, add new date)
    - Edit Data Layer Properties (Description, add new date)
    - Edit Map Doc Properties (Description, add new date)
  - Republish service(s)
    - In Service Editor → Parameters → Output Directory change Directory to "proxy output"

FOR FUTURE REFERENCE ONLY, services are not currently being cached.

- *IF CACHING:*
    - *Pooling -> Set Maximum number of instances to **4** (only set one to 4 at a time)*
    - *Under Caching, Set "Using tiles from a cache"*
        - *Accept all Levels of Detail to full extent (the actual levels to be cached will be changed later)*
        - *Select "Update cache manually".*
        - *Under Advanced Caching*
            - *Make sure minimum cached scale is set to largest tile size, and make sure maximum cached scale is set to smallest tile size.*
            - *Change Cache directory to D:\arcgisserver\directories\proxycache*
            - *Area of interest to cache: Import from feature class, navigate to Database connection for Database*

Connections\NPSDATA.DBO.INPNISCVDBNRSST_IMDDATA.s de and select IMD_BOUNDS_WEBMERCATOR_CONVEXHULL

***NOTE**: if you can't see anything in the connection dialogue, go back to ArcCatalog and preview the feature class in that program, then go back into the Manage Tiles dialogue and try again.*

- *Make sure CREATE TILES ON DEMAND is checked*
- *Analyze, then Publish (if no ERRORS)*

Literature Cited

Nelson, Lisa and Michelle Kinseth. 2015. NPS I&M Standard Operating Procedure: Inventory and Monitoring Boundary Processing. NPS/NRSS/IMD/SOP-1.0.

Appendices

Appendix 1: Known issues

Data currency

NPS boundary data are continually updated by the NPS Land Resources Division (Lands). The current NPScape NPS boundary-derived AOAs are based on the NPS Current Administrative Boundary dataset generated by Lands. All NPScape AOAs (custom and standard) are updated **ANNUALLY**. Any LANDS corrections/updates/etc. that occur after will not be reflected until next update, if they are relevant to the NPScape project.

Source Data Quality

NPScape AOAs are not meant to be an official source for park boundaries. They are only applicable to the NPScape project and the processing of Landscape Dynamics within the various I&M Networks.

The NPS Land Resources Division notes which NPS boundary polygons are legacy (i.e., not updated) by populating the GIS_NOTES and CREATED_BY attribute in the Current Administrative Boundaries shapefile. If an AOA was created from a custom park boundary, that information will also be present in the attribute table, along with the source for the spatial data.

Appendix 2: Landscape Dynamics/NPScape Standard Areas of Analysis

The standard AOAs exist for most NPS Inventory and Monitoring Program parks. This defined subset is listed in a landscape dynamics/NPScape data service. Refer to it for more details:

http://irmadevservices.nps.gov/NPScapeStatistics/odata/ParksNetworksRegions

Appendix 3: Scripts

FOR REFERENCE ONLY

```
#####################################################################
#
#   Create_NewNPScapeAOAs.pyt
#
#   Uses post-processed NPS boundaries feature class
(GDB_NRSS.DBO.nps_bound_gen_simple_albers)
#   to create new file GDB/feature datasets for park, 3km, and 30km AOAs
#
#   Prerequisites/Inputs:
#       ArcGIS 10.2.2 or higher
```

```
#       Source post-processed NPScape Unit Boundaries feature class
#       XML metadata template in known subfolder (NPScape/Templates/Metadata)
#       Output Folder/Workspace
#
#   Outputs:
#       File geodatabases with NPScape AOA feature classes,
#       organized into feature datasets (CONUS, Alaska, Hawaii,
Pacific_Saipan_Guam, Pacific_Samoa)
#
#
#   Created by:  NPS Inventory and Monitoring Division GIS Program
#   Update date: 20150910
#
#
#####################################################################

#import modules
import arcpy
import os, time
from datetime import datetime

# Class name must remain Toolbox to be recognized in ArcGIS
class Toolbox(object):
    def __init__(self):
        """Define the toolbox (the name of the toolbox is the name of the
        .pyt file)."""
        self.label = "Create_NewNPScapeAOAs"
        self.alias = ""

        # List of tool classes associated with this toolbox
        self.tools = [CreateServiceAOAs,CreateNewAOAs]

class CreateServiceAOAs(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Create Map Service NPScape AOAs"
        self.description = "Update NPScape AOAs with new IMD/NPScape unit
boundaries, AND creates AOAs for use with Map Services"
        self.canRunInBackground = False

    def getParameterInfo(self):
        """Define parameter definitions"""
        param0 = arcpy.Parameter(
            displayName = "Source NPScape Unit Bounds Feature Class",
            name = "unitBounds",
            datatype = "GPFeatureLayer",
            parameterType = "Required",
            direction = "Input")
        param0.value =
r"X:\ProjectData\Data_Processing\Bounds_Processing\Data\NPScape_Bounds.gdb\np
scape_bounds_albers"

        param1 = arcpy.Parameter(
            displayName = "Output Folder",
            name = "outFolder",
            datatype = "DEFolder",
```

```
            parameterType = "Required",
            direction = "Input")

        param2 = arcpy.Parameter(
            displayName = "Unit and Network Lookup Table",
            name = "Lookup",
            datatype = "Table",
            parameterType = "Required",
            direction = "Input")
        #param2.value = "Database
Connections\\NPScape_Stats.DBO_INPNISCVDBNRSST_IMDGIS.sde\\NPScape_Stats.dbo.
ParksNetworksRegions"

        params = [param0,param1,param2]
        return params

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        return

    def updateMessages(self, parameters):
        """Modify the messages created by internal validation for each tool
        parameter.  This method is called after internal validation."""
        return

    def createFeatureClass(self, OutputSelect, OutputGDB, FeatureClass,
UnitLookup, Lookup, StatusLookup, MetadataTemplate): #, FeatureDataset2,
FeatureDataset,
        """
            Create Feature Classes for CONUS, AK, Pacific Islands
        """
        arcpy.CopyFeatures_management(OutputSelect, OutputGDB
+"\\"+FeatureClass)
        arcpy.SelectLayerByAttribute_management(OutputSelect,
"CLEAR_SELECTION", "")
        arcpy.RepairGeometry_management(OutputGDB+"\\"+FeatureClass)
        #arcpy.CopyRows_management(UnitLookup, Lookup); arcpy.AddMessage('\n
Copy Lookup')

arcpy.JoinField_management(OutputGDB+"\\"+FeatureClass,"UNIT_CODE",Lookup,"Un
itCode",["UnitName","NetworkCode","NetworkName","RegionName"])

arcpy.JoinField_management(OutputGDB+"\\"+FeatureClass,"UNIT_CODE",StatusLook
up,"UNIT_CODE",["DATE_EDIT","Feature_Source","Source_Metadata"])
        arcpy.MetadataImporter_conversion(MetadataTemplate,
OutputGDB+"\\"+FeatureClass) #; arcpy.AddMessage('\n MEtdata')
        arcpy.AddMessage ('Created ' + OutputGDB + 'source feature class')

    def populateFeatureClass(self, OutputSelect, OutputGDB, fd,
buffFeatureClass, BuffDist, UnitLookup):
```

```
        """
            Populate Feature Class with spatial data
        """
        arcpy.Buffer_analysis(OutputSelect, OutputGDB + "\\"+fd+"\\" +
buffFeatureClass, BuffDist, "FULL", "ROUND", "NONE")
        arcpy.RepairGeometry_management(OutputGDB + "\\"+fd+"\\" +
buffFeatureClass)
        #arcpy.JoinField_management(OutputGDB + "\\"+fd+"\\" +
buffFeatureClass,"UNIT_CODE",UnitLookup,"UnitCode","UnitName")
        arcpy.AddField_management(OutputGDB + "\\"+fd+"\\" +
buffFeatureClass, "AREA_SQKM", "FLOAT", "11", "3", "", "", "NULLABLE",
"NON_REQUIRED", "")
        arcpy.CalculateField_management(OutputGDB + "\\"+fd+"\\" +
buffFeatureClass, "AREA_SQKM", "!shape.area! / 1000000", "PYTHON", "") #;
arcpy.AddMessage('n\ calc field')

    def execute(self, parameters, messages):
        message = ""
        today=datetime.now()
        datestamp = str(today.isoformat()).replace('-','')[0:8]

        arcpy.env.overwriteOutput = 1
        Output_FeatureClass_Root = "npscape_bounds_albers"  # Gets created
and copied into each CONUS GDB
        Source_FeatureClass = parameters[0].valueAsText # Source feature
class (our NPScape-only post-processed version of Lands latest bounds)
        Dissolved_FeatureClass_Root = "NPScape_Bounds_Albers_Dissolved"
        Dissolved_FC_3km = 'NPScape_3km_Dissolved'
        Dissolved_FC_30km = 'NPScape_30km_Dissolved'

        # Use a copy of the enteprise GDB feature class (our post-processed
version of Lands latest) as the source
        Output_Workspace = parameters[1].valueAsText
        Output_TargetGDB =
os.path.join(parameters[1].valueAsText,"NPScape_AOA.gdb")
        Versioned_Output_TargetGDB = os.path.join(parameters[1].valueAsText,
"NPScape_AOA_" + datestamp + ".gdb")
        Output_FeatureClass_30km = "AOA_30km"
        Output_FeatureClass_3km = "AOA_3km"
        Output_FeatureClass_Park = "AOA_Park"
        Output_Service_Park = "Park"

        BufferDistance = "30 Kilometers"
        BufferDistance_3km = "3 Kilometers"

        Unit_Lookups = parameters[2].valueAsText
        Lookup = Output_TargetGDB + "\\Lookup"
        Status_Lookups = Source_FeatureClass
        Metadata_Template =
r'X:\ProjectData\NPScape\Templates\Metadata\NPScape_AOA_Template.xml'

        ServiceSpatialRef =
'PROJCS["WGS_1984_Web_Mercator_Auxiliary_Sphere",GEOGCS["GCS_WGS_1984",DATUM[
"D_WGS_1984",SPHEROID["WGS_1984",6378137.0,298.257223563]],PRIMEM["Greenwich"
,0.0],UNIT["Degree",0.017453292519943295]],PROJECTION["Mercator_Auxiliary_Sph
ere"],PARAMETER["False_Easting",0.0],PARAMETER["False_Northing",0.0],PARAMETE
```

```
R["Central_Meridian",0.0],PARAMETER["Standard_Parallel_1",0.0],PARAMETER["Aux
iliary_Sphere_Type",0.0],UNIT["Meter",1.0]]'
        ConusSpatialRef =
'PROJCS["USA_Contiguous_Albers_Equal_Area_Conic_USGS_version",GEOGCS["GCS_Nor
th_American_1983",DATUM["D_North_American_1983",SPHEROID["GRS_1980",6378137.0
,298.257222101]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]],P
ROJECTION["Albers"],PARAMETER["False_Easting",0.0],PARAMETER["False_Northing"
,0.0],PARAMETER["Central_Meridian",-
96.0],PARAMETER["Standard_Parallel_1",29.5],PARAMETER["Standard_Parallel_2",4
5.5],PARAMETER["Latitude_Of_Origin",23.0],UNIT["Meter",1.0]]",
"NAD_1983_To_WGS_1984_4",
"GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHEROID["WGS_1984",6378137.0,298.2
57223563]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]]")'
#ConusSRDesc.SpatialReference
        AKSpatialRef =
'PROJCS["Alaska_Albers_Equal_Area_Conic",GEOGCS["GCS_North_American_1983",DAT
UM["D_North_American_1983",SPHEROID["GRS_1980",6378137,298.257222101]],PRIMEM
["Greenwich",0],UNIT["Degree",0.017453292519943295]],PROJECTION["Albers"],PAR
AMETER["False_Easting",0],PARAMETER["False_Northing",0],PARAMETER["central_me
ridian",-
154],PARAMETER["Standard_Parallel_1",55],PARAMETER["Standard_Parallel_2",65],
PARAMETER["latitude_of_origin",50],UNIT["Meter",1]]'
#AKSRDesc.SpatialReference
        HawaiiSpatialRef =
'PROJCS["NAD_1983_UTM_Zone_5N",GEOGCS["GCS_North_American_1983",DATUM["D_Nort
h_American_1983",SPHEROID["GRS_1980",6378137,298.257222101]],PRIMEM["Greenwic
h",0],UNIT["Degree",0.017453292519943295]],PROJECTION["Transverse_Mercator"],
PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",-
153],PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAM
ETER["false_northing",0],UNIT["Meter",1]]' #HawaiiSRDesc.SpatialReference
        PacificSaipanSpatialRef =
'PROJCS["WGS_1984_UTM_Zone_55N",GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHE
ROID["WGS_1984",6378137,298.257223563]],PRIMEM["Greenwich",0],UNIT["Degree",0
.017453292519943295]],PROJECTION["Transverse_Mercator"],PARAMETER["latitude_o
f_origin",0],PARAMETER["central_meridian",147],PARAMETER["scale_factor",0.999
6],PARAMETER["false_easting",500000],PARAMETER["false_northing",0],UNIT["Mete
r",1]]' #PacificSaipanSRDesc.SpatialReference
        transformMethodSaipan = "WGS_1984_(ITRF00)_To_NAD_1983"  # ????? Use
custom PacP00 transform/custom projection to change to NAD83 datum:
WGS_1984_To_NAD_1983_PACP00.gtf
        PacificSamoaSpatialRef = 'PROJCS["NAD83(HARN) / UTM zone
2S",GEOGCS["NAD83(HARN)",DATUM["D_North_American_1983_HARN",SPHEROID["GRS_198
0",6378137,298.257222101]],PRIMEM["Greenwich",0],UNIT["Degree",0.017453292519
943295]],PROJECTION["Transverse_Mercator"],PARAMETER["latitude_of_origin",0],
PARAMETER["central_meridian",-
171],PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAM
ETER["false_northing",10000000],UNIT["Meter",1]]'
#PacificSamoaSRDesc.SpatialReference

        Conus_FeatureDataset_Park = "CONUS_Park"
        AK_FeatureDataset_Park = "Alaska_Park"
        Hawaii_FeatureDataset_Park = "Hawaii_Park"
        PacificSaipan_FeatureDataset_Park = "Pacific_Saipan_Guam_Park"
        PacificSamoa_FeatureDataset_Park = "Pacific_Samoa_Park"
        Conus_FeatureDataset_3km = "CONUS_3km"
        AK_FeatureDataset_3km = "Alaska_3km"
```

```
        Hawaii_FeatureDataset_3km = "Hawaii_3km"
        PacificSaipan_FeatureDataset_3km = "Pacific_Saipan_Guam_3km"
        PacificSamoa_FeatureDataset_3km = "Pacific_Samoa_3km"
        Conus_FeatureDataset_30km = "CONUS_30km"
        AK_FeatureDataset_30km = "Alaska_30km"
        Hawaii_FeatureDataset_30km = "Hawaii_30km"
        PacificSaipan_FeatureDataset_30km = "Pacific_Saipan_Guam_30km"
        PacificSamoa_FeatureDataset_30km = "Pacific_Samoa_30km"

        ServiceSourceFeatureDataset = "ServiceSourcePark"

        Service_FeatureClass = "NPScape_Dissolved"
        AK_FeatureClass = "NPS_Alaska_Dissolved"
        Hawaii_FeatureClass = "NPS_Hawaii_Dissolved"
        PacificSaipan_FeatureClass = "NPS_Pacific_Saipan_Guam_Dissolved"
        PacificSamoa_FeatureClass = "NPS_Pacific_Samoa_Dissolved"
        Conus_FeatureClass = "NPS_CONUS_Dissolved"

        AllUnits = [
            'ABLI', 'ACAD', 'AGFO', 'ALAG', 'ALFL', 'ALKA', 'ALPO', 'AMIS',
'AMME', 'ANAC', 'ANIA', 'ANTI', 'APPA', 'APCO', 'APIS', 'ARCH', 'ARPO',
'ASIS', 'AZRU',
            'BADL', 'BAND', 'BAWA', 'BELA', 'BEOL', 'BIBE', 'BICA', 'BICY',
'BIHO', 'BISC', 'BISO', 'BITH', 'BLCA', 'BLRI', 'BLUE', 'BOHA', 'BOWA',
            'BRCA', 'BUFF', 'BUIS', 'CABR', 'CACH', 'CACO', 'CAGR', 'CAHA',
'CAHE', 'CAKR', 'CALO', 'CANA', 'CANY', 'CARE', 'CARL', 'CASA', 'CATO',
            'CAVE', 'CAVO', 'CEBR', 'CHAT', 'CHCH', 'CHCU', 'CHIC', 'CHIR',
'CHIS', 'CHOH', 'CHPI', 'CIRO', 'COLM', 'COLO', 'CORO', 'CONG', 'COWP',
            'CRLA', 'CRMO', 'CUGA', 'CUIS', 'CURE', 'CUVA', 'DENA', 'DEPO',
'DETO', 'DEVA', 'DEWA', 'DINO', 'DRTO', 'EBLA', 'EFMO', 'EISE', 'ELMA',
            'ELMO', 'EUON', 'EVER', 'FIIS', 'FLFO', 'FOBO', 'FOBU', 'FOCA',
'FODA', 'FODO', 'FOFR', 'FOLA', 'FOLS', 'FOMA', 'FOMO', 'FONE',
            'FOPO', 'FOPU', 'FORA', 'FOSU', 'FOUN', 'FOUS', 'FOVA', 'FOWA',
'FRDO', 'FRHI', 'FRSP', 'GAAR', 'GARI', 'GATE', 'GETT', 'GEWA', 'GICL',
            'GLAC', 'GLBA', 'GLCA', 'GOGA', 'GOSP', 'GRBA', 'GRCA', 'GREE',
'GRKO', 'GRPO', 'GRSA', 'GRSM', 'GRTE', 'GUCO', 'GUIS', 'GUMO', 'GWCA',
            'GWMP', 'HAFE', 'HAFO', 'HALE', 'HAVO', 'HEHO', 'HOBE', 'HOCU',
'HOFU', 'HOME', 'HOSP', 'HOVE', 'HUTR', 'INDU', 'ISRO',
            'JECA', 'JELA', 'JODA', 'JOFL', 'JOMU', 'JOTR', 'KAHO', 'KALA',
'KATM', 'KEFJ', 'KEMO', 'KIMO', 'KLGO', 'KNRI', 'KOVA', 'LABE',
            'LACL', 'LAKE', 'LAMR', 'LARO', 'LAVO', 'LEWI', 'LIBI', 'LIBO',
'LIRI', 'LYJO', 'MABE', 'MABI', 'MACA', 'MANA', 'MANZ', 'MEVE', 'MIMA',
            'MISS', 'MNRR', 'MOCA', 'MOCR', 'MOJA', 'MONO', 'MORA', 'MORR',
'MORU', 'MUWO', 'NABR', 'NACE', 'NATR', 'NAVA', 'NEPE', 'NERI', 'NIOB',
            'NISI', 'NOAT', 'NOCA', 'NPSA', 'OBRI', 'OCMU', 'OLYM', 'ORCA',
'ORPI', 'OZAR', 'PAAL', 'PAIS', 'PARA', 'PECO', 'PEFO', 'PERI', 'PETE',
'PETR',
            'PINN', 'PIPE', 'PIRO', 'PISC', 'PISP', 'POCH', 'PORE', 'PRSF',
'PRWI', 'PUHE', 'PUHO', 'RABR', 'REDW', 'RICH', 'RIGR', 'ROCR', 'ROMO',
            'ROVA', 'RUCA', 'SAAN', 'SACN', 'SAGA', 'SAGU', 'SAHI', 'SAIR',
'SAJH', 'SAMO', 'SAND', 'SAPU', 'SARA', 'SCBL', 'SEKI',
            'SHEN', 'SHIL', 'SITK', 'SLBE', 'STRI', 'SUCR', 'TAPR', 'THRO',
'THST', 'TICA', 'TIMU', 'TONT', 'TUMA', 'TUZI', 'UPDE', 'VALR', 'VAFO',
            'VICK', 'VIIS', 'VOYA', 'WABA', 'WACA', 'WAPA', 'WEFA', 'WHIS',
'WHMI', 'WHSA', 'WICA', 'WICR', 'WOTR', 'WRBR', 'WRST', 'WUPA',
            'YELL', 'YOSE', 'YUCH', 'YUHO', 'ZION']
```

```
        Alaska_Parks =
"'ALAG','ANIA','BELA','CAKR','DENA','GAAR','GLBA','KATM','KEFJ','KOVA','KLGO'
,'LACL','NOAT','SITK','YUCH','WRST'"
        Alaska_Parks_Array =
['ALAG','ANIA','BELA','CAKR','DENA','GAAR','GLBA','KATM','KEFJ','KOVA','KLGO'
,'LACL','NOAT','SITK','YUCH','WRST']
        Hawaii_Parks =
"'ALKA','HALE','KALA','KAHO','HAVO','PUHE','PUHO','VALR'"
        Hawaii_Parks_Array =
['ALKA','HALE','KALA','KAHO','HAVO','PUHE','PUHO','VALR']
        PacificSaipan_Parks = "'AMME','WAPA'"
        PacificSaipan_Parks_Array = ['AMME','WAPA']
        PacificSamoa_Parks= "'NPSA'"
        PacificSamoa_Parks_Array = ['NPSA']
        Conus_Select_Array0 =
[Alaska_Parks_Array,Hawaii_Parks_Array,PacificSaipan_Parks_Array,PacificSamoa
_Parks_Array]
        Conus_Select_Array = [ y for x in Conus_Select_Array0 for y in x]
        for unit in Conus_Select_Array:
            AllUnits.remove(unit)

        # All NPS units under NPScape - change to more elegant array
        ServiceParksArray = [
            'ABLI', 'ACAD', 'AGFO','ALAG', 'ALFL', 'ALKA', 'ALPO', 'AMIS',
'AMME', 'ANAC', 'ANIA', 'ANTI', 'APPA', 'APCO', 'APIS', 'ARCH', 'ARPO',
'ASIS', 'AZRU',
            'BADL', 'BAND', 'BAWA', 'BELA', 'BEOL', 'BIBE', 'BICA', 'BICY',
'BIHO', 'BISC', 'BISO', 'BITH', 'BLCA', 'BLRI', 'BLUE', 'BOHA', 'BOWA',
            'BRCA', 'BUFF', 'BUIS', 'CABR', 'CACH', 'CACO', 'CAGR', 'CAHA',
'CAHE', 'CAKR', 'CALO', 'CANA', 'CANY', 'CARE', 'CARL', 'CASA', 'CATO',
            'CAVE', 'CAVO', 'CEBR', 'CHAT', 'CHCH', 'CHCU', 'CHIC', 'CHIR',
'CHIS', 'CHOH', 'CHPI', 'CIRO', 'COLM', 'COLO', 'CORO', 'CONG', 'COWP',
            'CRLA', 'CRMO', 'CUGA', 'CUIS', 'CURE', 'CUVA', 'DENA', 'DEPO',
'DETO', 'DEVA', 'DEWA', 'DINO', 'DRTO', 'EBLA', 'EFMO', 'EISE', 'ELMA',
            'ELMO', 'EUON', 'EVER', 'FIIS', 'FLFO', 'FOBO', 'FOBU', 'FOCA',
'FODA', 'FODO', 'FOFR', 'FOLA', 'FOLS', 'FOMA', 'FOMO', 'FONE',
            'FOPO', 'FOPU', 'FORA', 'FOSU', 'FOUN', 'FOUS', 'FOVA', 'FOWA',
'FRDO', 'FRHI', 'FRSP', 'GAAR', 'GARI', 'GATE', 'GETT', 'GEWA', 'GICL',
            'GLAC', 'GLBA', 'GLCA', 'GOGA', 'GOSP', 'GRBA', 'GRCA', 'GREE',
'GRKO', 'GRPO', 'GRSA', 'GRSM', 'GRTE', 'GUCO', 'GUIS', 'GUMO', 'GWCA',
            'GWMP', 'HAFE', 'HAFO', 'HALE', 'HAVO', 'HEHO', 'HOBE', 'HOCU',
'HOFU', 'HOME', 'HOSP', 'HOVE', 'HUTR', 'INDU', 'ISRO',
            'JECA', 'JELA', 'JODA', 'JOFL', 'JOMU', 'JOTR', 'KAHO', 'KALA',
'KATM', 'KEFJ', 'KEMO', 'KIMO', 'KLGO', 'KNRI', 'KOVA', 'LABE',
            'LACL', 'LAKE', 'LAMR', 'LARO', 'LAVO', 'LEWI', 'LIBI', 'LIBO',
'LIRI', 'LYJO', 'MABE', 'MABI', 'MACA', 'MANA', 'MANZ', 'MEVE', 'MIMA',
            'MISS', 'MNRR', 'MOCA', 'MOCR', 'MOJA', 'MONO', 'MORA', 'MORR',
'MORU', 'MUWO', 'NABR', 'NACE', 'NATR', 'NAVA', 'NEPE', 'NERI', 'NIOB',
            'NISI', 'NOAT', 'NOCA', 'NPSA', 'OBRI', 'OCMU', 'OLYM', 'ORCA',
'ORPI', 'OZAR', 'PAAL', 'PAIS', 'PARA', 'PECO', 'PEFO', 'PERI', 'PETE',
'PETR',
            'PINN', 'PIPE', 'PIRO', 'PISC', 'PISP', 'POCH', 'PORE', 'PRSF',
'PRWI', 'PUHE', 'PUHO', 'RABR', 'REDW', 'RICH', 'RIGR', 'ROCR', 'ROMO',
```

```
            'ROVA', 'RUCA', 'SAAN', 'SACN', 'SAGA', 'SAGU', 'SAHI', 'SAIR',
'SAJH', 'SAMO', 'SAND', 'SAPU', 'SARA', 'SCBL', 'SEKI',
            'SHEN', 'SHIL', 'SITK', 'SLBE', 'STRI', 'SUCR', 'TAPR', 'THRO',
'THST', 'TICA', 'TIMU', 'TONT', 'TUMA', 'TUZI', 'UPDE', 'VALR', 'VAFO',
            'VICK', 'VIIS', 'VOYA', 'WABA', 'WACA', 'WAPA', 'WEFA', 'WHIS',
'WHMI', 'WHSA', 'WICA', 'WICR', 'WOTR', 'WRBR', 'WRST', 'WUPA',
            'YELL', 'YOSE', 'YUCH', 'YUHO', 'ZION']


        Output_Layer = "AOABounds_Layer"
        Output_Select = "AOABounds_Select"
        OutputService = "NPScape_Dissolved"

        OutputsToDelete = [Output_Layer, Output_Select]

        for output in OutputsToDelete:
            if arcpy.Exists(output):
                arcpy.Delete_management(output); showGpMessage()

        try:
            if arcpy.Exists(Versioned_Output_TargetGDB):
                arcpy.Delete_management(Versioned_Output_TargetGDB);
messages.addGPMessages()
            if arcpy.Exists(Output_TargetGDB):
                arcpy.Rename_management(Output_TargetGDB,
Versioned_Output_TargetGDB); messages.addGPMessages()

            arcpy.CreateFileGDB_management(Output_Workspace,
"NPScape_AOA.gdb"); messages.addGPMessages()

            # Copy NPScape boundary feature class into new file geodatabases
            arcpy.MakeFeatureLayer_management(Source_FeatureClass,
Output_Layer); messages.addGPMessages()
            arcpy.CopyFeatures_management(Output_Layer, Output_TargetGDB +
"\\" + Output_FeatureClass_Root); messages.addGPMessages()
            arcpy.Delete_management(Output_Layer); messages.addGPMessages()

            arcpy.RepairGeometry_management(Output_TargetGDB + "\\" +
Output_FeatureClass_Root); messages.addGPMessages()
            arcpy.MetadataImporter_conversion(Metadata_Template,
Output_TargetGDB + "\\" + Output_FeatureClass_Root); arcpy.AddMessage('\n
Created AOA GDB and source feature class \n') #; messages.addGPMessages()

            # Dissolve NPS boundary feature class on UNIT_CODE
            arcpy.MakeFeatureLayer_management(Output_TargetGDB + "\\" +
Output_FeatureClass_Root, Output_Layer); messages.addGPMessages()
            arcpy.Dissolve_management(Output_Layer,Output_TargetGDB +"\\" +
Dissolved_FeatureClass_Root,"UNIT_CODE"); messages.addGPMessages()
            arcpy.RepairGeometry_management(Output_TargetGDB +"\\" +
Dissolved_FeatureClass_Root); messages.addGPMessages()
            arcpy.Delete_management(Output_Layer); messages.addGPMessages()
            arcpy.Delete_management(Output_TargetGDB + "\\" +
Output_FeatureClass_Root); messages.addGPMessages()
            arcpy.MetadataImporter_conversion(Metadata_Template,
Output_TargetGDB + "\\" + Dissolved_FeatureClass_Root); arcpy.AddMessage('\n
Created GDB and source feature class \n') #; messages.addGPMessages()
```

```
            # For park-based GDBs, add and populate Feature Datasets for
CONUS (includes PR and VI), Alaska and Pacific Islands

arcpy.CreateFeatureDataset_management(Output_TargetGDB,ServiceSourceFeatureDa
taset,ServiceSpatialRef); messages.addGPMessages()


arcpy.CreateFeatureDataset_management(Output_TargetGDB,Conus_FeatureDataset_3
0km,ConusSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,AK_FeatureDataset_30km
,AKSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,Hawaii_FeatureDataset_
30km,HawaiiSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,PacificSaipan_FeatureD
ataset_30km,PacificSaipanSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,PacificSamoa_FeatureDa
taset_30km,PacificSamoaSpatialRef); messages.addGPMessages()


arcpy.CreateFeatureDataset_management(Output_TargetGDB,Conus_FeatureDataset_3
km,ConusSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,AK_FeatureDataset_3km,
AKSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,Hawaii_FeatureDataset_
3km,HawaiiSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,PacificSaipan_FeatureD
ataset_3km,PacificSaipanSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,PacificSamoa_FeatureDa
taset_3km,PacificSamoaSpatialRef); messages.addGPMessages()


arcpy.CreateFeatureDataset_management(Output_TargetGDB,Conus_FeatureDataset_P
ark,ConusSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,AK_FeatureDataset_Park
,AKSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,Hawaii_FeatureDataset_
Park,HawaiiSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,PacificSaipan_FeatureD
ataset_Park,PacificSaipanSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,PacificSamoa_FeatureDa
taset_Park,PacificSamoaSpatialRef); messages.addGPMessages()

            # Populate the Web Service Source Feature Dataset
```

```
            arcpy.MakeFeatureLayer_management(Output_TargetGDB +"\\" +
Dissolved_FeatureClass_Root, Output_Select); messages.addGPMessages()
            arcpy.CopyFeatures_management(Output_Select, Output_TargetGDB +
"\\"+ServiceSourceFeatureDataset+"\\" + Service_FeatureClass);
messages.addGPMessages()
            arcpy.RepairGeometry_management(Output_TargetGDB +
"\\"+ServiceSourceFeatureDataset+"\\" + Service_FeatureClass);
messages.addGPMessages()
            arcpy.CopyRows_management(Unit_Lookups, Lookup);
messages.addGPMessages()
            arcpy.JoinField_management(Output_TargetGDB +
"\\"+ServiceSourceFeatureDataset+"\\" +
Service_FeatureClass,"UNIT_CODE",Lookup,"UnitCode",["UnitName","NetworkCode",
"NetworkName","RegionName"]); messages.addGPMessages()
            arcpy.JoinField_management(Output_TargetGDB +
"\\"+ServiceSourceFeatureDataset+"\\" +
Service_FeatureClass,"UNIT_CODE",Status_Lookups,"UNIT_CODE",["DATE_EDIT","Fea
ture_Source","Source_Metadata"]); messages.addGPMessages()
            FieldCalc = ("!UNIT_CODE!+'_'+'"+Output_FeatureClass_Park+"'")
            arcpy.AddField_management(Output_TargetGDB + "\\"
+ServiceSourceFeatureDataset+"\\"+ Service_FeatureClass, "AOA_NAME",
"TEXT","","", "50", "AOA_NAME","NULLABLE","REQUIRED","");
arcpy.AddMessage('\n Add Field AOA_NAME Park Service \n')
            arcpy.CalculateField_management(Output_TargetGDB + "\\"
+ServiceSourceFeatureDataset+"\\"+ Service_FeatureClass, "AOA_NAME",
FieldCalc,"PYTHON_9.3"); arcpy.AddMessage('\n Calculate Field AOA_NAME Park
Service \n')
            #arcpy.MetadataImporter_conversion(MetadataTemplate,
Output_TargetGDB + "\\"+ServiceSourceFeatureDataset+"\\"
+Service_FeatureClass)

            # Populate CONUS, AK, HI, Saipan, Samoa Feature Classes
            arcpy.SelectLayerByAttribute_management(Output_Select,
"NEW_SELECTION", "UNIT_CODE IN ("+Alaska_Parks+")"); messages.addGPMessages()
            arcpy.env.outputCoordinateSystem = AKSpatialRef
            CreateServiceAOAs.createFeatureClass(self, Output_Select,
Output_TargetGDB, AK_FeatureClass, Unit_Lookups, Lookup, Status_Lookups,
Metadata_Template)
            arcpy.ClearEnvironment("outputCoordinateSystem")

            arcpy.SelectLayerByAttribute_management(Output_Select,
"NEW_SELECTION", "UNIT_CODE IN ("+Hawaii_Parks+")"); messages.addGPMessages()
            arcpy.env.outputCoordinateSystem = HawaiiSpatialRef
            CreateServiceAOAs.createFeatureClass(self, Output_Select,
Output_TargetGDB, Hawaii_FeatureClass, Unit_Lookups, Lookup, Status_Lookups,
Metadata_Template)
            arcpy.ClearEnvironment("outputCoordinateSystem")

            arcpy.SelectLayerByAttribute_management(Output_Select,
"NEW_SELECTION", "UNIT_CODE IN ("+PacificSamoa_Parks+")");
messages.addGPMessages()
            arcpy.env.outputCoordinateSystem = PacificSamoaSpatialRef
            CreateServiceAOAs.createFeatureClass(self, Output_Select,
Output_TargetGDB, PacificSamoa_FeatureClass, Unit_Lookups, Lookup,
Status_Lookups, Metadata_Template)
            arcpy.ClearEnvironment("outputCoordinateSystem")
```

```
            arcpy.SelectLayerByAttribute_management(Output_Select,
"NEW_SELECTION", "UNIT_CODE IN ("+Alaska_Parks+")"); messages.addGPMessages()
            arcpy.SelectLayerByAttribute_management(Output_Select,
"ADD_TO_SELECTION", "UNIT_CODE IN ("+PacificSaipan_Parks+")");
messages.addGPMessages()
            arcpy.SelectLayerByAttribute_management(Output_Select,
"ADD_TO_SELECTION", "UNIT_CODE IN ("+PacificSamoa_Parks+")");
messages.addGPMessages()
            arcpy.SelectLayerByAttribute_management(Output_Select,
"ADD_TO_SELECTION", "UNIT_CODE IN ("+Hawaii_Parks+")");
messages.addGPMessages()
            arcpy.SelectLayerByAttribute_management(Output_Select,
"SWITCH_SELECTION"); messages.addGPMessages()
            CreateServiceAOAs.createFeatureClass(self, Output_Select,
Output_TargetGDB, Conus_FeatureClass, Unit_Lookups, Lookup, Status_Lookups,
Metadata_Template)
            arcpy.ClearEnvironment("outputCoordinateSystem")

            arcpy.SelectLayerByAttribute_management(Output_Select,
"NEW_SELECTION", "UNIT_CODE IN ("+PacificSaipan_Parks+")");
messages.addGPMessages()
            arcpy.env.outputCoordinateSystem = PacificSaipanSpatialRef
            arcpy.env.geographicTransformations = (transformMethodSaipan);
messages.addGPMessages()
            CreateServiceAOAs.createFeatureClass(self, Output_Select,
Output_TargetGDB, PacificSaipan_FeatureClass, Unit_Lookups, Lookup,
Status_Lookups, Metadata_Template)
            arcpy.ClearEnvironment("geographicTransformations")
            arcpy.ClearEnvironment("outputCoordinateSystem")

            arcpy.SelectLayerByAttribute_management(Output_Select,
"CLEAR_SELECTION", ""); messages.addGPMessages()

            # For each feature class by UNIT_CODE, select feature, copy to
Output_TargetGDB_Park, then buffer to 3km and 30km for
Output_TargetGDB_3km/30km
            arcpy.env.workspace = Output_TargetGDB

            fds = arcpy.ListDatasets("*Park","All"); arcpy.AddMessage(fds)

            for fd in fds:
                if fd == 'CONUS_Park':
                    park_List = AllUnits  # CONUS feature dataset
                    park_fc = Conus_FeatureClass
                    fd30 = Conus_FeatureDataset_30km
                    fd3 = Conus_FeatureDataset_3km
                elif fd == 'Alaska_Park':
                    park_List = Alaska_Parks_Array
                    park_fc = AK_FeatureClass
                    fd30 = AK_FeatureDataset_30km
                    fd3 = AK_FeatureDataset_3km
                elif fd == 'Hawaii_Park':
                    park_List = Hawaii_Parks_Array
                    park_fc = Hawaii_FeatureClass
                    fd30 = Hawaii_FeatureDataset_30km
```

```
                    fd3 = Hawaii_FeatureDataset_3km
                elif fd == 'Pacific_Saipan_Guam_Park':
                    park_List = PacificSaipan_Parks_Array
                    park_fc = PacificSaipan_FeatureClass
                    fd30 = PacificSaipan_FeatureDataset_30km
                    fd3 = PacificSaipan_FeatureDataset_3km
                elif fd == 'Pacific_Samoa_Park':
                    park_List = PacificSamoa_Parks_Array
                    park_fc = PacificSamoa_FeatureClass
                    fd30 = PacificSamoa_FeatureDataset_30km
                    fd3 = PacificSamoa_FeatureDataset_3km
                elif fd == 'ServiceSourcePark':
                    park_List = ServiceParksArray
                    park_fc = OutputService
                else:
                    pass

                try:
                    for park in park_List:
                        print arcpy.GetMessages()
                        message = "\n\nERROR: Buffering: " + park
                        buffered_FeatureClass = park + "_" +
Output_FeatureClass_30km
                        buffered_FeatureClass_3km = park + "_" +
Output_FeatureClass_3km
                        park_FeatureClass = park + "_" +
Output_FeatureClass_Park

arcpy.MakeFeatureLayer_management(park_fc,Output_Select);
messages.addGPMessages()

arcpy.SelectLayerByAttribute_management(Output_Select,"NEW_SELECTION","UNIT_C
ODE = \'"+park+"\'"); messages.addGPMessages()

                        if fd == 'ServiceSourcePark':
                            park_FeatureClass = park + "_" +
Output_Service_Park
                        else:
                            pass


arcpy.CopyFeatures_management(Output_Select,Output_TargetGDB + "\\" + fd +
"\\" + park_FeatureClass); messages.addGPMessages()
                        ##arcpy.JoinField_management(Output_TargetGDB + "\\"
+ fd + "\\" +
park_FeatureClass,"UNIT_CODE",Unit_Lookups,"UnitCode",["UnitName","NetworkCod
e","NetworkName","RegionName"]); messages.addGPMessages()
                        arcpy.AddField_management(Output_TargetGDB + "\\" +
fd + "\\" + park_FeatureClass, "AREA_SQKM", "FLOAT", "11", "3", "", "",
"NULLABLE", "NON_REQUIRED", ""); messages.addGPMessages()
                        arcpy.CalculateField_management(Output_TargetGDB +
"\\" + fd + "\\" + park_FeatureClass, "AREA_SQKM", "!shape.area! / 1000000",
"PYTHON", ""); messages.addGPMessages()

                        if fd != 'ServiceSourcePark':
```

```
CreateServiceAOAs.populateFeatureClass(self,Output_Select,Output_TargetGDB,fd
30,buffered_FeatureClass,BufferDistance,Unit_Lookups)

CreateServiceAOAs.populateFeatureClass(self,Output_Select,Output_TargetGDB,fd
3,buffered_FeatureClass_3km,BufferDistance_3km,Unit_Lookups)

arcpy.MetadataImporter_conversion(Metadata_Template, Output_TargetGDB + "\\"
+ fd30 + "\\" + buffered_FeatureClass); messages.addGPMessages()

arcpy.MetadataImporter_conversion(Metadata_Template, Output_TargetGDB + "\\"
+ fd3 + "\\" + buffered_FeatureClass_3km); messages.addGPMessages()
                        else:
                            pass

                        #arcpy.MetadataImporter_conversion(Metadata_Template,
Output_TargetGDB + "\\" + fd + "\\" + park_FeatureClass);
messages.addGPMessages()

arcpy.SelectLayerByAttribute_management(Output_Select, "CLEAR_SELECTION",
""); messages.addGPMessages()


                except:
                    print arcpy.GetMessages()
                    message = "\n\nERROR: Cannot extract or buffer features
in: " + park_List
                    raise Exception, message

            # Create complete feature classes of AOAs for use in Stats Tools
            Alaska_3km = Output_TargetGDB +
"\\"+AK_FeatureDataset_3km+"\\Alaska_" + Output_FeatureClass_3km
            Alaska_30km = Output_TargetGDB +
"\\"+AK_FeatureDataset_30km+"\\Alaska_" + Output_FeatureClass_30km
            CONUS_3km = Output_TargetGDB +
"\\"+Conus_FeatureDataset_3km+"\\CONUS_" + Output_FeatureClass_3km
            CONUS_30km = Output_TargetGDB +
"\\"+Conus_FeatureDataset_30km+"\\CONUS_" + Output_FeatureClass_30km
            Hawaii_3km = Output_TargetGDB +
"\\"+Hawaii_FeatureDataset_3km+"\\Hawaii_" + Output_FeatureClass_3km
            Hawaii_30km = Output_TargetGDB +
"\\"+Hawaii_FeatureDataset_30km+"\\Hawaii_" + Output_FeatureClass_30km
            GuamSaipan_3km = Output_TargetGDB +
"\\"+PacificSaipan_FeatureDataset_3km+"\\Saipan_" + Output_FeatureClass_3km
            GuamSaipan_30km = Output_TargetGDB +
"\\"+PacificSaipan_FeatureDataset_30km+"\\Saipan_" + Output_FeatureClass_30km
            Samoa_3km = Output_TargetGDB +
"\\"+PacificSamoa_FeatureDataset_3km+"\\Samoa_" + Output_FeatureClass_3km
            Samoa_30km = Output_TargetGDB +
"\\"+PacificSamoa_FeatureDataset_30km+"\\Samoa_" + Output_FeatureClass_30km
            arcpy.Buffer_analysis(Output_TargetGDB + "\\" +
AK_FeatureClass,Alaska_3km,BufferDistance_3km,"FULL","ROUND","NONE");
messages.addGPMessages()
            arcpy.Buffer_analysis(Output_TargetGDB + "\\" +
AK_FeatureClass,Alaska_30km,BufferDistance,"FULL","ROUND","NONE");
arcpy.AddMessage('\n Created web mercator dissolved feature classes \n')
```

```
arcpy.CopyFeatures_management(AK_FeatureClass,Output_TargetGDB+"\\"+
AK_FeatureDataset_Park +"\\"+ "Alaska_AOA_Park"); arcpy.AddMessage('\n Copy
Alaska \n')
            arcpy.Buffer_analysis(Output_TargetGDB + "\\" +
Conus_FeatureClass,CONUS_3km,BufferDistance_3km,"FULL","ROUND","NONE");
messages.addGPMessages()
            arcpy.Buffer_analysis(Output_TargetGDB + "\\" +
Conus_FeatureClass,CONUS_30km,BufferDistance,"FULL","ROUND","NONE");
arcpy.AddMessage('\n Created web mercator dissolved feature classes \n')

arcpy.CopyFeatures_management(Conus_FeatureClass,Output_TargetGDB+"\\"+Conus_
FeatureDataset_Park+"\\" + "Conus_AOA_Park"); arcpy.AddMessage('\n Copy Conus
\n')
            arcpy.Buffer_analysis(Output_TargetGDB + "\\" +
Hawaii_FeatureClass,Hawaii_3km,BufferDistance_3km,"FULL","ROUND","NONE");
messages.addGPMessages()
            arcpy.Buffer_analysis(Output_TargetGDB + "\\" +
Hawaii_FeatureClass,Hawaii_30km,BufferDistance,"FULL","ROUND","NONE");
arcpy.AddMessage('\n Created web mercator dissolved feature classes \n')

arcpy.CopyFeatures_management(Hawaii_FeatureClass,Output_TargetGDB
+"\\"+Hawaii_FeatureDataset_Park +"\\"+ "Hawaii_AOA_Park");
arcpy.AddMessage('\n Copy Hawaii \n')
            arcpy.Buffer_analysis(Output_TargetGDB + "\\" +
PacificSaipan_FeatureClass,GuamSaipan_3km,BufferDistance_3km,"FULL","ROUND","
NONE"); messages.addGPMessages()
            arcpy.Buffer_analysis(Output_TargetGDB + "\\" +
PacificSaipan_FeatureClass,GuamSaipan_30km,BufferDistance,"FULL","ROUND","NON
E"); arcpy.AddMessage('\n Created web mercator dissolved feature classes \n')

arcpy.CopyFeatures_management(PacificSaipan_FeatureClass,Output_TargetGDB+"\\
"+PacificSaipan_FeatureDataset_Park+"\\" + "Saipan_AOA_Park");
arcpy.AddMessage('\n Copy Saipan \n')
            arcpy.Buffer_analysis(Output_TargetGDB + "\\" +
PacificSamoa_FeatureClass,Samoa_3km,BufferDistance_3km,"FULL","ROUND","NONE")
; messages.addGPMessages()
            arcpy.Buffer_analysis(Output_TargetGDB + "\\" +
PacificSamoa_FeatureClass,Samoa_30km,BufferDistance,"FULL","ROUND","NONE");
arcpy.AddMessage('\n Created web mercator dissolved feature classes \n')

arcpy.CopyFeatures_management(PacificSamoa_FeatureClass,Output_TargetGDB+"\\"
+PacificSamoa_FeatureDataset_Park+"\\" + "Samoa_AOA_Park");
arcpy.AddMessage('\n Copy Samoa \n')

            arcpy.AddMessage('\n Add field, copy service features to SDE \n')

            # Create Web Mercator dissolved feature classes for 3km and 30km
buffered layers
            Service3km_Dissolve = Output_TargetGDB +
"\\"+ServiceSourceFeatureDataset+"\\" + "NPScape_Dissolved_3km"
            Service30km_Dissolve = Output_TargetGDB +
"\\"+ServiceSourceFeatureDataset+"\\" + "NPScape_Dissolved_30km"
            arcpy.Buffer_analysis(Output_TargetGDB + "\\"
+ServiceSourceFeatureDataset+"\\"+
```

```
Service_FeatureClass,Service3km_Dissolve,BufferDistance_3km,"FULL","ROUND","N
ONE"); messages.addGPMessages()
            arcpy.CalculateField_management(Service3km_Dissolve, "AOA_NAME",
"!UNIT_CODE!+'_AOA_3km'","PYTHON_9.3"); arcpy.AddMessage('\n Calculate Field
AOA_NAME 3km Service \n')
            arcpy.Buffer_analysis(Output_TargetGDB + "\\"
+ServiceSourceFeatureDataset+"\\"+
Service_FeatureClass,Service30km_Dissolve,BufferDistance,"FULL","ROUND","NONE
"); arcpy.AddMessage('\n Created web mercator dissolved feature classes \n')
            arcpy.CalculateField_management(Service30km_Dissolve, "AOA_NAME",
"!UNIT_CODE!+'_AOA_30km'","PYTHON_9.3"); arcpy.AddMessage('\n Calculate Field
AOA_NAME 30km Service \n')

            # Copy Web Mercator feature classes to SDE

arcpy.CopyFeatures_management(Output_TargetGDB+"\\"+ServiceSourceFeatureDatas
et+"\\"+Service_FeatureClass, "Database
Connections\\GDB_NRSS.DBO.INPNISCVDBNRSST_IMDGIS.sde\\" +
"NPScape_Park_webmercator"); messages.addGPMessages()
            arcpy.CopyFeatures_management(Service3km_Dissolve, "Database
Connections\\GDB_NRSS.DBO.INPNISCVDBNRSST_IMDGIS.sde\\" +
"NPScape_3km_webmercator"); messages.addGPMessages()
            arcpy.CopyFeatures_management(Service30km_Dissolve, "Database
Connections\\GDB_NRSS.DBO.INPNISCVDBNRSST_IMDGIS.sde\\" +
"NPScape_30km_webmercator"); arcpy.AddMessage('\n Copied web mercator
dissolved feature classes to SDE \n')

            arcpy.Delete_management(Lookup); messages.addGPMessages()

        except:
            print arcpy.GetMessages()
            message = "\n\nERROR: Cannot create AOA feature classes
correctly."
            raise Exception, message
```

DRAFT