



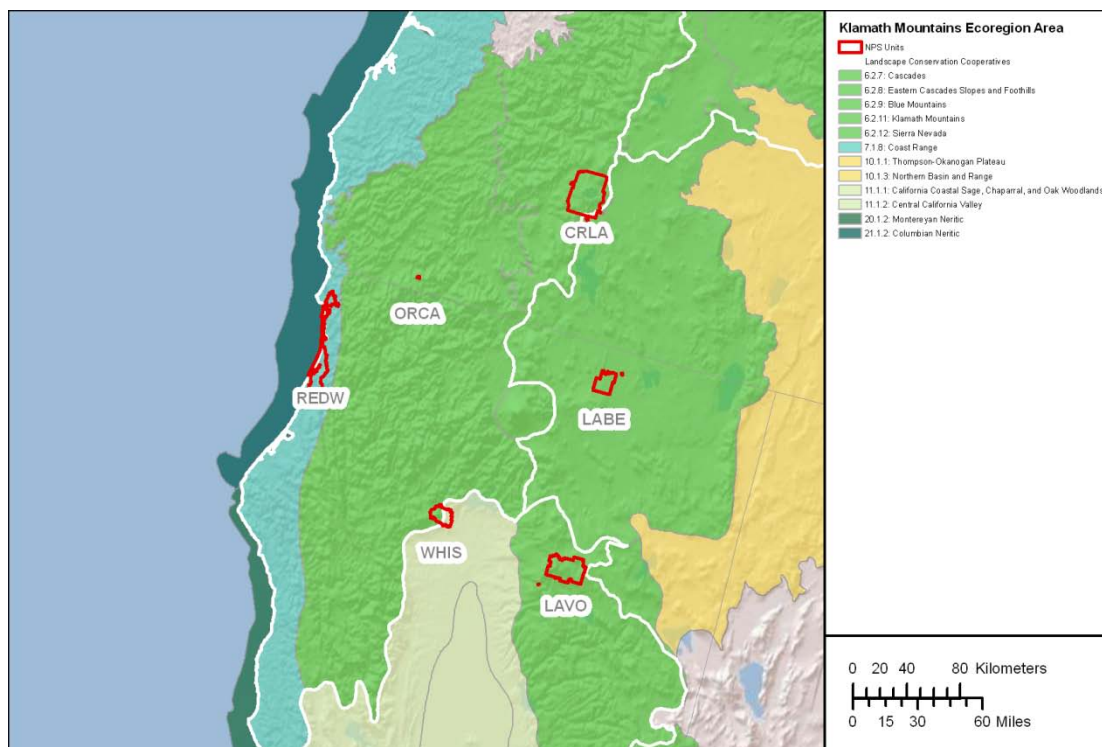
NOTE: This document reflects the processes used to generate the spring 2011 release of NPScape data. There may be revised processes and documentation available.

Check Reference Application (<http://nriminfo.nps.gov>) for most current version.

NPScape Ecoregion-Derived Areas of Analysis SOP

*Commission for Environmental Cooperation Terrestrial and
Marine Ecoregions and FWS Landscape Conservation
Cooperative Areas of Analysis*

Version: 20110411



Suggested Citation: National Park Service. 2011. NPScape ecoregion-derived areas of analysis SOP: Commission for Environmental Cooperation terrestrial and marine ecoregions and FWS Landscape Conservation Cooperative areas of analysis. National Park Service, Natural Resource Program Center. Fort Collins, Colorado.

Table of Contents

1.	Overview.....	3
2.	Data Acquisition and Preprocessing.....	3
2.1.	Source Data.....	3
2.2.	Data Re-Projection.....	4
3.	AOA Processing	4
3.1.	AOA Python Scripting.....	4
4.	Quality Control	4
4.1.	Verify Values for Calculated Areas	4
4.2.	Check Geometries	4
5.	Appendices	5
5.1.	Known Issues	5
5.2.	Processing Scripts	5

1. Overview

This NPScape SOP documents how areas of analysis (AOAs) were derived from ecoregion boundaries for use in NPScape analyses. Three ecoregion-based AOAs are used in NPScape: Commission on Environmental Cooperation (CEC) terrestrial and marine ecoregions and FWS Landscape Conservation Cooperatives (LCC). The AOA polygons are derived by extracting ecoregion polygons from these source datasets. An area attribute (AREA_SQKM) is calculated for each AOA polygon. See Appendix 5.2 for details.

Python scripts are used to create these AOAs as feature classes within feature datasets in one of three standardized file geodatabases: NPScape_AOA_CEC.gdb (terrestrial CEC ecoregions), NPScape_AOA_Marine_CEC.gdb, and NPScape_AOA_LCC.gdb.



These AOA file geodatabases, plus a lookup database for NPScape park lists, parks within ecoregions, park geospatial shape indexes, and other details may be downloaded from the NPScape web site:

<http://science.nature.nps.gov/im/monitor/npscape/>

The data sources and tools used are assumed to be compatible with ESRI ArcGIS™ format, version 9.3.1 Service Pack 1 or version 10 Service Pack 1.

2. Data Acquisition and Preprocessing

2.1. Source Data

CEC and FWS LCC source datasets were obtained from the respective agencies.

- **Source 1:** CEC Terrestrial and Marine Ecoregions (downloaded in August 2010)

<http://www.cec.org/naatlas>

- **Source 2:** FWS Landscape Conservation Cooperatives (downloaded in August 2010)

http://www.fws.gov/GIS/data/national/FWS_LCC.zip

2.2. Data Re-Projection

Each source dataset must be re-projected into a common spatial reference. For CONUS areas (including Puerto Rico and the Virgin Islands), the NPScape project uses USA Contiguous Albers Equal Area Conic USGS as its standard projection. For Alaska, Alaska Albers Equal Area Conic is used. NAD_83 is the datum for both projections. For Hawaii, UTM Zone 5N NAD83 is used. The NPScape AOA polygons are available in feature datasets matching these projections in the respective file geodatabases.

The ecoregion boundary source data were re-projected into the above spatial references as part of the AOA processing.

3. AOA Processing

3.1. AOA Python Scripting

Following feature correction, a suite of Python scripts are used to generate AOA polygons from ecoregion boundaries. See Appendix 5.2 for scripts.

General processing steps of the tools are:

Create file geodatabase, create feature dataset with correct spatial reference; copy ecoregion polygons; repair geometry; dissolve source feature class on ecoregion name attribute; iterate through each dissolved ecoregion polygon, copying it into a new feature class in the relevant feature dataset; add AREA_SQKM attribute and calculate the area in square kilometers; import metadata; merge individual AOA polygons into a merged AOA feature class

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

NAME, NAME_L3 or area_names = the ecoregion name

AREA_SQKM: total area of the AOA polygon in square kilometers

4. Quality Control

4.1. Verify Values for Calculated Areas

Using ArcMap, the AREA_SQKM values are spot-checked for a selected set of ecoregion-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.

4.2. Check Geometries

Using ArcMap, the merged AOA feature class is overlaid with the source ecoregion boundary polygons to check for spatial congruency.

5. Appendices

5.1. Known Issues

Data Currency

Source data were downloaded in August 2010 and represent ecoregion boundaries as of that date.

5.2. Processing Scripts

Create NewNPScape_CEC_AOAs.py

```
try:
    # Import system modules
    import sys, string, os, glob, datetime, arcgisscripting
    from datetime import datetime
    message = ""
    arcGISRoot = ""

    # Create the Geoprocessor object
    gp = arcgisscripting.create(9.3)

    gp.OverwriteOutput = 1

    #Messaging functions (copied from Chris Snyder's source)
    def showGpMessage():
        try:
            print "\n" + gp.GetMessages() + "\n"
        except:
            pass
    def showPyMessage():
        try:
            print str(time.ctime()) + " - " + str(message)
        except:
            pass

    # Local variables...
    today=datetime.now()
    datestamp = str(today.isoformat()).replace('-',')[0:8]
    print datestamp

    Scratch_Workspace = "E:\\Scratch\\ProcessingOutputs"
    NPScape_Workspace = "E:\\NPScape\\Scratch"
    # For each feature class by CEC Name, select feature, copy to Output_TargetGDB_CEC, then buffer to
    10km

    CECSource_Folder =
    "O:\\SourceData\\NorthAmericaData\\Boundary\\CEC_TerrestrialEcoregions_Level3\\CEC_TerrestrialEcore
    gions_L3.gdb"
    CECSource_FeatureClass_Root = "Terrestrial_Ecoregions_Level3_Albers"
    CECSource_FeatureClass = CECSource_Folder + "\\ " + CECSource_FeatureClass_Root
    Dissolved_FeatureClass_Root = "CEC_Albers_Dissolved"

    Output_TargetRoot = "NPScape_AOA"
```

```
Output_TargetGDB_CEC = NPScape_Workspace + "\\" + Output_TargetRoot + "_CEC.gdb"
Versioned_Output_TargetGDB = NPScape_Workspace + "\\" + Output_TargetRoot + "_CEC_" + datestamp
+ ".gdb"
```

```
Output_FeatureClass_CEC = "AOA_CEC"
Output_FeatureClass_Buffered_CEC = "AOA_CEC_10km"
```

```
BufferDistance = "10 Kilometers"
```

```
Metadata_Template = "E:\\NPScape\\Templates\\Metadata\\NPScape_AOA_Template.xml"
```

```
AKSRSource = "O:\\SourceData\\USData\\Boundary\\AK_NPS_Boundary.gdb\\AK_Albers"
AKSRDesc = gp.Describe(AKSRSource)
AKSpatialRef = AKSRDesc.SpatialReference
#HawaiiSRSource =
"O:\\SourceData\\USData\\Boundary\\Hawaii_Coastline\\Hawaii_Coastline.gdb\\hawai_envelope"
#HawaiiSRDesc = gp.Describe(HawaiiSRSource)
#HawaiiSpatialRef = HawaiiSRDesc.SpatialReference
```

```
AK_FeatureDataset = "Alaska"
#Hawaii_FeatureDataset = "Hawaii"
Conus_FeatureDataset = "CONUS"
AK_FeatureClass = "CEC_Alaska"
#Hawaii_FeatureClass = "CEC_Hawaii"
#Hawaii_FeatureClass_Temp = "CEC_Hawaii_Temp"
Conus_FeatureClass = "CEC_CONUS"
```

```
CONUS_CECs = "'Algonquin Southern Laurentians','Arizona New Mexico Mountains','Arizona New Mexico Plateau', 'Arkansas Valley','Aspen Parkland Northern Glaciated Plains','Atlantic Coastal Pine Barrens','Blue Mountains', 'Blue Ridge','Boston Mountains','California Coastal Sage, Chaparral, and Oak Woodlands','Canadian Rockies','Cascades', 'Central Appalachians','Central Basin and Range','Central California Valley','Central Corn Belt Plains','Central Great Plains','Central Irregular Plains','Chihuahuan Desert','Coast Range','Colorado Plateaus','Columbia Mountains Northern Rockies', 'Columbia Plateau','Cross Timbers','Driftless Area','East Central Texas Plains','Eastern Cascades Slopes and Foothills', 'Eastern Corn Belt Plains','Eastern Great Lakes and Hudson Lowlands','Edwards Plateau','Erie Drift Plain','Flint Hills', 'High Plains','Huron Erie Lake Plains','Idaho Batholith','Interior Plateau','Interior River Valleys and Hills','Klamath Mountains', 'Lake Erie Lowland','Lake Manitoba and Lake Agassiz Plain','Madrean Archipelago','Maine New Brunswick Plains and Hills', 'Middle Atlantic Coastal Plain','Middle Rockies','Mississippi Alluvial Plain','Mississippi Valley Loess Plains', 'Mojave Basin and Range','Nebraska Sand Hills','North Cascades','North Central Appalachians','North Central Hardwood Forests', 'Northeastern Coastal Zone','Northern Appalachian and Atlantic Maritime Highlands','Northern Appalachian Plateau and Uplands', 'Northern Basin and Range','Northern Lakes and Forests','Northern Minnesota Wetlands','Northern Piedmont', 'Northwestern Glaciated Plains','Northwestern Great Plains','Ouachita Mountains','Ozark Highlands','Piedmont', 'Ridge and Valley','South Michigan North Indiana Drift Plains','Sierra Nevada','Snake River Plain','Sonoran Desert', 'South Central Plains','Southeastern Plains','Southeastern Wisconsin Till Plains', 'Southern and Baja California Pine Oak Mountains','Southern Coastal Plain','Southern Florida Coastal Plain', 'Southern Rockies','Southern Texas Plains Interior Plains and Hills with Xerophytic Shrub and Oak Forest', 'Southwestern Appalachians','Southwestern Tablelands','Strait of Georgia Puget Lowland','Texas Blackland Prairies', 'Wasatch and Uinta Mountains','Western Allegheny Plateau','Western Corn Belt Plains','Western Gulf Coastal Plain', 'Willamette Valley','Wyoming Basin'"
```

```
CONUS_CEC_Array = ['Algonquin Southern Laurentians','Arizona New Mexico Mountains','Arizona New Mexico Plateau','Arkansas Valley',
, 'Aspen Parkland Northern Glaciated Plains','Atlantic Coastal Pine Barrens','Blue Mountains','Blue Ridge','Boston Mountains'
, 'California Coastal Sage Chaparral and Oak Woodlands','Canadian Rockies','Cascades','Central Appalachians'
, 'Central Basin and Range','Central California Valley','Central Corn Belt Plains','Central Great Plains'
```

, 'Central Irregular Plains', 'Chihuahuan Desert', 'Coast Range', 'Colorado Plateaus', 'Columbia Mountains Northern Rockies'
 , 'Columbia Plateau', 'Cross Timbers', 'Driftless Area', 'East Central Texas Plains', 'Eastern Cascades Slopes and Foothills'
 , 'Eastern Corn Belt Plains', 'Eastern Great Lakes and Hudson Lowlands', 'Edwards Plateau', 'Erie Drift Plain', 'Flint Hills'
 , 'High Plains', 'Huron Erie Lake Plains', 'Idaho Batholith', 'Interior Plateau', 'Interior River Valleys and Hills', 'Klamath Mountains'
 , 'Lake Erie Lowland', 'Lake Manitoba and Lake Agassiz Plain', 'Madrean Archipelago', 'Maine New Brunswick Plains and Hills'
 , 'Middle Atlantic Coastal Plain', 'Middle Rockies', 'Mississippi Alluvial Plain', 'Mississippi Valley Loess Plains'
 , 'Mojave Basin and Range', 'Nebraska Sand Hills', 'North Cascades', 'North Central Appalachians', 'North Central Hardwood Forests'
 , 'Northeastern Coastal Zone', 'Northern Appalachian and Atlantic Maritime Highlands', 'Northern Appalachian Plateau and Uplands'
 , 'Northern Basin and Range', 'Northern Lakes and Forests', 'Northern Minnesota Wetlands', 'Northern Piedmont'
 , 'Northwestern Glaciated Plains', 'Northwestern Great Plains', 'Ouachita Mountains', 'Ozark Highlands', 'Piedmont'
 , 'Ridge and Valley', 'South Michigan North Indiana Drift Plains', 'Sierra Nevada', 'Snake River Plain', 'Sonoran Desert'
 , 'South Central Plains', 'Southeastern Plains', 'Southeastern Wisconsin Till Plains'
 , 'Southern and Baja California Pine Oak Mountains', 'Southern Coastal Plain', 'Southern Florida Coastal Plain'
 , 'Southern Rockies', 'Southern Texas Plains Interior Plains and Hills with Xerophytic Shrub and Oak Forest'
 , 'Southwestern Appalachians', 'Southwestern Tablelands', 'Strait of Georgia Puget Lowland', 'Texas Blackland Prairies'
 , 'Wasatch and Uinta Mountains', 'Western Allegheny Plateau', 'Western Corn Belt Plains', 'Western Gulf Coastal Plain'
 , 'Willamette Valley', 'Wyoming Basin']

Alaska_CECs = ["Ahklun and Kilbuck Mountains", 'Alaska Peninsula Mountains', 'Alaska Range', 'Aleutian Islands', 'Arctic Coastal Plain', 'Arctic Foothills', 'Bristol Bay Nushagak Lowlands', 'Brooks Range Richardson Mountains', 'Coastal Western Hemlock Sitka Spruce Forests', 'Cook Inlet', 'Copper Plateau', 'Interior Bottomlands', 'Interior Forested Lowlands and Uplands', 'Interior Highlands and Klondike Plateau', 'Ogilvie Mountains', 'Pacific Coastal Mountains', 'Seward Peninsula', 'Subarctic Coastal Plains', 'Wrangell and St Elias Mountains', 'Yukon Flats', 'Yukon Stikine Highlands Boreal Mountains and Plateaus']

Alaska_CEC_Array = ['Ahklun and Kilbuck Mountains', 'Alaska Peninsula Mountains', 'Alaska Range', 'Aleutian Islands', 'Arctic Coastal Plain', 'Arctic Foothills', 'Bristol Bay Nushagak Lowlands', 'Brooks Range Richardson Mountains', 'Coastal Western Hemlock Sitka Spruce Forests', 'Cook Inlet', 'Copper Plateau', 'Interior Bottomlands', 'Interior Forested Lowlands and Uplands', 'Interior Highlands and Klondike Plateau', 'Ogilvie Mountains', 'Pacific Coastal Mountains', 'Seward Peninsula', 'Subarctic Coastal Plains', 'Wrangell and St Elias Mountains', 'Yukon Flats', 'Yukon Stikine Highlands Boreal Mountains and Plateaus']

Output_Layer = "AOABounds_Layer"
 Output_Select = "AOABounds_Select"
 CEC_Layer = "CEC_Layer"

gp.ScratchWorkspace = Scratch_Workspace
 gp.workspace = NPScape_Workspace

OutputsToDelete = [Versioned_Output_TargetGDB, Output_Layer, Output_Select, CEC_Layer]

for output in OutputsToDelete:
 if gp.Exists(output):

```

gp.Delete(output); showGpMessage()

try:
    if gp.Exists(Output_TargetGDB_CEC):
        gp.Rename_management(Output_TargetGDB_CEC, Versioned_Output_TargetGDB);
showGpMessage()
    #gp.Delete_management(Output_TargetGDB_CEC); showGpMessage()

    gp.CreateFileGDB_management(NPScape_Workspace, Output_TargetRoot + "_CEC.gdb");
showGpMessage()

    # Copy CEC feature class into new file geodatabases
    gp.Copy_management(CECSource_FeatureClass, Output_TargetGDB_CEC + "\\" +
CECSource_FeatureClass_Root); showGpMessage()

    gp.RepairGeometry_management(Output_TargetGDB_CEC + "\\" + CECSource_FeatureClass_Root);
showGpMessage()

    ConusSRSource = Output_TargetGDB_CEC + "\\" + CECSource_FeatureClass_Root
    ConusSRDesc = gp.Describe(ConusSRSource); showGpMessage()
    ConusSpatialRef = ConusSRDesc.SpatialReference

    # Dissolve CEC feature class on [NAME] (terrestrial CECs)
    gp.MakeFeatureLayer_management(Output_TargetGDB_CEC + "\\" + CECSource_FeatureClass_Root,
Output_Layer); showGpMessage()
    gp.Dissolve_management(Output_Layer,Output_TargetGDB_CEC + "\\" +
Dissolved_FeatureClass_Root,"NAME"); showGpMessage()
    gp.RepairGeometry_management(Output_TargetGDB_CEC + "\\" + Dissolved_FeatureClass_Root);
showGpMessage()
    gp.Delete_management(Output_Layer); showGpMessage()
    gp.Delete_management(Output_TargetGDB_CEC + "\\" + CECSource_FeatureClass_Root);
showGpMessage()
    print 'Created CEC AOA GDB and source feature class'

    # For CEC GDB, add and populate Feature Datasets for CONUS (includes PR and VI), Alaska and Hawaii
(marine only)

gp.CreateFeatureDataset_management(Output_TargetGDB_CEC,Conus_FeatureDataset,ConusSpatialRef);
showGpMessage()
    gp.CreateFeatureDataset_management(Output_TargetGDB_CEC,AK_FeatureDataset,AKSpatialRef);
showGpMessage()

#gp.CreateFeatureDataset_management(Output_TargetGDB_CEC,Hawaii_FeatureDataset,PacificSpatialRef)
; showGpMessage()

    # Get features from dissolved source
    gp.MakeFeatureLayer_management(Output_TargetGDB_CEC + "\\" + Dissolved_FeatureClass_Root,
Output_Select); showGpMessage()

    gp.SelectLayerByAttribute_management(Output_Select, "NEW_SELECTION", "NAME IN
("+Alaska_CECs+")"); showGpMessage()
    gp.CopyFeatures_management(Output_Select, Output_TargetGDB_CEC + "\\"+AK_FeatureDataset+"\"
+ AK_FeatureClass); showGpMessage()
    gp.SelectLayerByAttribute_management(Output_Select, "CLEAR_SELECTION", ""); showGpMessage()
    gp.RepairGeometry_management(Output_TargetGDB_CEC + "\\"+AK_FeatureDataset+"\" +
AK_FeatureClass); showGpMessage()

```

```

    #gp.SelectLayerByAttribute_management(Output_Select, "NEW_SELECTION", "NAME_L3 IN
    ("+"Hawaii_CECs+"")); showGpMessage()
    #gp.CopyFeatures_management(Output_Select, Output_TargetGDB_CEC + "\\" +
    Hawaii_FeatureDataset + "\\" + Hawaii_FeatureClass); showGpMessage()
    #gp.SelectLayerByAttribute_management(Output_Select, "CLEAR_SELECTION", ""); showGpMessage()
    #gp.RepairGeometry_management(Output_TargetGDB_CEC + "\\" + Hawaii_FeatureDataset + "\\" +
    Hawaii_FeatureClass); showGpMessage()

    gp.SelectLayerByAttribute_management(Output_Select, "NEW_SELECTION", "NAME IN
    ("+"Alaska_CECs+"")); showGpMessage()
    #gp.SelectLayerByAttribute_management(Output_Select, "ADD_TO_SELECTION", "NAME_L3 IN
    ("+"Pacific_CECs+"")); showGpMessage()
    gp.SelectLayerByAttribute_management(Output_Select, "SWITCH_SELECTION"); showGpMessage()
    gp.CopyFeatures_management(Output_Select, Output_TargetGDB_CEC +
    "\\" + Conus_FeatureDataset + "\\" + Conus_FeatureClass); showGpMessage()
    gp.SelectLayerByAttribute_management(Output_Select, "CLEAR_SELECTION", ""); showGpMessage()
    gp.RepairGeometry_management(Output_TargetGDB_CEC + "\\" + Conus_FeatureDataset + "\\" +
    Conus_FeatureClass); showGpMessage()

    gp.workspace = Output_TargetGDB_CEC
    fds = gp.ListDatasets("", "Feature"); showGpMessage()

    for fd in fds:
        if fd == 'Alaska':
            cec_List = Alaska_CEC_Array
        elif fd == 'Hawaii':
            cec_List = Hawaii_CEC_Array
        else: cec_List = CONUS_CEC_Array
        #else cec_List = [s for s in Conus_Parks.split()]
        gp.SelectLayerByAttribute(Output_Select, "CLEAR_SELECTION", ""); showGpMessage()

    try:
        print "Extracting and buffering " + fd + " CECs"
        for cec in cec_List:
            buffered_FeatureClass = cec.replace(' ', '_') + "_" + Output_FeatureClass_Buffered_CEC
            cec_FeatureClass = cec.replace(' ', '_') + "_" + Output_FeatureClass_CEC
            gp.SelectLayerByAttribute_management(Output_Select, "NEW_SELECTION", "NAME =
            \"+" + cec + "+\""); showGpMessage()
            gp.CopyFeatures_management(Output_Select, Output_TargetGDB_CEC + "\\" + fd + "\\" +
            cec_FeatureClass); showGpMessage()
            gp.AddField_management(Output_TargetGDB_CEC + "\\" + fd + "\\" + cec_FeatureClass,
            "AREA_SQKM", "FLOAT", "", "", "", "NULLABLE", "NON_REQUIRED", ""); showGpMessage()
            gp.CalculateField_management(Output_TargetGDB_CEC + "\\" + fd + "\\" + cec_FeatureClass,
            "AREA_SQKM", "!shape.area! / 1000000", "PYTHON_9.3", ""); showGpMessage()
            gp.MetadataImporter(Metadata_Template, Output_TargetGDB_CEC + "\\" + fd + "\\" +
            cec_FeatureClass); showGpMessage()
            gp.Buffer_analysis(Output_Select, Output_TargetGDB_CEC + "\\" + fd + "\\" +
            buffered_FeatureClass, BufferDistance, "FULL", "ROUND", "NONE"); showGpMessage()
            gp.RepairGeometry_management(Output_TargetGDB_CEC + "\\" + fd + "\\" +
            buffered_FeatureClass); showGpMessage()
            gp.AddField_management(Output_TargetGDB_CEC + "\\" + fd + "\\" + buffered_FeatureClass,
            "AREA_SQKM", "FLOAT", "", "", "", "NULLABLE", "NON_REQUIRED", ""); showGpMessage()
            gp.CalculateField_management(Output_TargetGDB_CEC + "\\" + fd + "\\" +
            buffered_FeatureClass, "AREA_SQKM", "!shape.area! / 1000000", "PYTHON_9.3", ""); showGpMessage()
            gp.SelectLayerByAttribute_management(Output_Select, "CLEAR_SELECTION", "");
            showGpMessage()
            gp.MetadataImporter(Metadata_Template, Output_TargetGDB_CEC + "\\" + fd + "\\" +
            cec_FeatureClass); showGpMessage()

```

```

        print "Completed " + cec
    except:
        print gp.getMessages()
        message = "\n\nERROR: Cannot buffer features in: " + cec_List
        raise Exception, message

    try:
        gp.workspace = fd
        merged_FeatureClass = fd + "_AOAs"
        fcList0 = ""
        fcs = gp.ListFeatureClasses("*_AOA_*", "POLYGON")
        for fc in fcs:
            fcList0 += Output_TargetGDB_CEC + "\\ " + fd + "\\ " + fc + ";"
        fcList = fcList0[0:-1] + ""
        gp.CreateFeatureClass(Output_TargetGDB_CEC + "\\ " +
fd,merged_FeatureClass,"POLYGON",buffered_FeatureClass); showGpMessage()
        gp.Append_management(fcList,Output_TargetGDB_CEC + "\\ " + fd + "\\ " +
merged_FeatureClass,"TEST"); showGpMessage()
        gp.RepairGeometry_management(Output_TargetGDB_CEC + "\\ " + fd + "\\ " +
merged_FeatureClass); showGpMessage()
        #gp.JoinField_management(Output_TargetGDB_CEC + "\\ " + fd + "\\ " +
merged_FeatureClass,"NAME",Unit_Lookups,"NAME","NAME"); showGpMessage()
        print 'Created merged feature class ' + merged_FeatureClass
    except:
        print gp.getMessages()
        message = "\n\nERROR: Cannot create merged feature class for " + fd
        raise Exception, message

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

message = "Finished creating new NPScape CEC AOA geodatabase."
#del gp
print message

except:
    print 'PROCESSING ERROR:\n'
    message = "\n*** LAST GEOPROCESSOR MESSAGE (may not be source of the error)***";
showPyMessage()
showGpMessage()
message = "\n*** PYTHON ERRORS *** "; showPyMessage()
message = "Python Traceback Info: " + traceback.format_tb(sys.exc_info()[2])[0]; showPyMessage()
message = "Python Error Info: " + str(sys.exc_type)+ ": " + str(sys.exc_value) + "\n"; showPyMessage()
message = "\n*** PYTHON LOCAL VARIABLE LIST ***"; showPyMessage()

```

Create NewNPScape MarineCEC AOAs.py

```
try:
    # Import system modules
    import sys, string, os, glob, datetime, arcgisscripting
    from datetime import datetime
    message = ""
    arcGISRoot = ""

    # Create the Geoprocessor object
    gp = arcgisscripting.create(9.3)

    gp.OverwriteOutput = 1

    #Messaging functions (copied from Chris Snyder's source)
    def showGpMessage():
        try:
            print "\n" + gp.GetMessages() + "\n"
        except:
            pass
    def showPyMessage():
        try:
            print str(time.ctime()) + " - " + str(message)
        except:
            pass

    # Local variables...
    today=datetime.now()
    datestamp = str(today.isoformat()).replace('-',')[0:8]
    print datestamp

    Scratch_Workspace = "E:\\Scratch\\ProcessingOutputs"
    NPScape_Workspace = "E:\\NPScape\\Scratch"
    # For each feature class by CEC Name, select feature, copy to Output_TargetGDB_CEC, then buffer to
    10km

    CECSource_Folder =
    "O:\\SourceData\\NorthAmericaData\\Boundary\\CEC_MarineEcoregions_Level3\\CEC_MarineEcoregions_
    L3.gdb"
    CECSource_FeatureClass_Root = "Marine_Ecoregions_Level3_Albers"
    CECSource_FeatureClass = CECSource_Folder + "\\\" + CECSource_FeatureClass_Root
    Dissolved_FeatureClass_Root = "CEC_Albers_Dissolved"

    Output_TargetRoot = "NPScape_AOA_Marine"
    Output_TargetGDB_CEC = NPScape_Workspace + "\\\" + Output_TargetRoot + "_CEC.gdb"
    Versioned_Output_TargetGDB = NPScape_Workspace + "\\\" + Output_TargetRoot + "_CEC_" + datestamp
    + ".gdb"

    Output_FeatureClass_CEC = "AOA_CEC"
    Output_FeatureClass_Buffered_CEC = "AOA_CEC_10km"

    BufferDistance = "10 Kilometers"

    Metadata_Template = "E:\\NPScape\\Templates\\Metadata\\NPScape_AOA_Template.xml"

    AKSRSource = "O:\\SourceData\\USData\\Boundary\\AK_NPS_Boundary.gdb\\AK_Albers"
    AKSRDesc = gp.Describe(AKSRSource)
    AKSpatialRef = AKSRDesc.SpatialReference
```

```
HawaiiSRSource =  
"O:\\SourceData\\USData\\Boundary\\Hawaii_Coastline\\Hawaii_Coastline.gdb\\hawai_envelope"  
HawaiiSRDesc = gp.Describe(HawaiiSRSource)  
HawaiiSpatialRef = HawaiiSRDesc.SpatialReference
```

```
AK_FeatureDataset = "Alaska"  
Hawaii_FeatureDataset = "Hawaii"  
Conus_FeatureDataset = "CONUS"  
AK_FeatureClass = "CEC_Alaska"  
Hawaii_FeatureClass = "CEC_Hawaii"  
Hawaii_FeatureClass_Temp = "CEC_Hawaii_Temp"  
Conus_FeatureClass = "CEC_CONUS"
```

```
Marine_CECs = "'Carolinian Neritic','Channel Islands Neritic','Columbia River Estuarine Area','Columbian  
Neritic','Delaware Chesapeake Bay','Dry Tortugas Florida Keys Reef Tract','Eastern Gulf Neritic','Ensenadan  
Neritic','Florida Bay','Florida Keys','Georges Bank Neritic','Gulf of Maine Bay of Fundy','Long Island Sound  
Buzzards Bay','Mississippi Estuarine Area','Montereyan Neritic','New York Bight','Pamlico Sound Outer  
Banks','Piedmont Estuarine Area','Puerto Rican Neritic','Puget Sound Estuarine Area','San Francisco Bay  
Estuarine Area','Scotian Neritic','Shark River Estuarine Area','Southeast Floridian Neritic','Southwest  
Floridian Neritic','St Croix Neritic','Strait of Georgia Estuarine Area','Strait of Juan de Fuca','Texas Estuarine  
Area','US Virgin Islands Viequean Neritic','Western Florida Estuarine Area','Western Gulf Neritic'"
```

```
Marine_CEC_Array = ['Carolinian Neritic','Channel Islands Neritic','Columbia River Estuarine  
Area','Columbian Neritic','Delaware Chesapeake Bay'  
, 'Dry Tortugas Florida Keys Reef Tract','Eastern Gulf Neritic','Ensenadan Neritic','Florida Bay','Florida  
Keys'  
, 'Georges Bank Neritic','Gulf of Maine Bay of Fundy','Long Island Sound Buzzards Bay','Mississippi  
Estuarine Area'  
, 'Montereyan Neritic','New York Bight','Pamlico Sound Outer Banks','Piedmont Estuarine Area','Puerto  
Rican Neritic'  
, 'Puget Sound Estuarine Area','San Francisco Bay Estuarine Area','Scotian Neritic','Shark River Estuarine  
Area'  
, 'Southeast Floridian Neritic','Southwest Floridian Neritic','St Croix Neritic','Strait of Georgia Estuarine  
Area'  
, 'Strait of Juan de Fuca','Texas Estuarine Area','US Virgin Islands Viequean Neritic','Western Florida  
Estuarine Area'  
, 'Western Gulf Neritic']
```

```
Alaska_CECs = "'Aleutian Neritic','Beaufortian Neritic','Bristol Bay and Kuskokwim Bay','Chukchian  
Neritic','Cook Inlet','Coronation Queen Maud Gulf','Fjordland Estuarine Areas','Fjordland Neritic','Gulf of  
Alaska','Kotzebue Sound','Mackenzie Estuarine Area','Middle and Outer Bering Sea Neritic','Puget Sound  
Estuarine Area','Strait of Georgia Estuarine Area','Strait of Juan de Fuca'"
```

```
Alaska_CEC_Array = ['Aleutian Neritic','Beaufortian Neritic','Bristol Bay and Kuskokwim Bay','Chukchian  
Neritic','Cook Inlet'  
, 'Coronation Queen Maud Gulf','Fjordland Estuarine Areas','Fjordland Neritic','Gulf of Alaska','Kotzebue  
Sound'  
, 'Mackenzie Estuarine Area','Middle and Outer Bering Sea Neritic']
```

```
Hawaii_CECs = "'Main Hawaiian Island Coastal','Northwestern Hawaiian Coastal'"
```

```
Hawaii_CEC_Array = ['Main Hawaiian Island Coastal','Northwestern Hawaiian Coastal']
```

```
Output_Layer = "AOABounds_Layer"  
Output_Select = "AOABounds_Select"  
CEC_Layer = "CEC_Layer"
```

```
gp.ScratchWorkspace = Scratch_Workspace
```

```

gp.workspace = NPScape_Workspace

OutputsToDelete = [Versioned_Output_TargetGDB, Output_Layer, Output_Select, CEC_Layer]

for output in OutputsToDelete:
    if gp.Exists(output):
        gp.Delete(output); showGpMessage()

try:
    if gp.Exists(Output_TargetGDB_CEC):
        gp.Rename_management(Output_TargetGDB_CEC, Versioned_Output_TargetGDB);
showGpMessage()
    #gp.Delete_management(Output_TargetGDB_CEC); showGpMessage()

    gp.CreateFileGDB_management(NPScape_Workspace, Output_TargetRoot + "_CEC.gdb");
showGpMessage()

    # Copy CEC feature class into new file geodatabases
    gp.Copy_management(CECSource_FeatureClass, Output_TargetGDB_CEC + "\\" +
CECSource_FeatureClass_Root); showGpMessage()

    gp.RepairGeometry_management(Output_TargetGDB_CEC + "\\" + CECSource_FeatureClass_Root);
showGpMessage()

    ConusSRSource = Output_TargetGDB_CEC + "\\" + CECSource_FeatureClass_Root
    ConusSRDesc = gp.Describe(ConusSRSource); showGpMessage()
    ConusSpatialRef = ConusSRDesc.SpatialReference

    # Dissolve CEC feature class on [NAME] (terrestrial CECs)
    gp.MakeFeatureLayer_management(Output_TargetGDB_CEC + "\\" + CECSource_FeatureClass_Root,
Output_Layer, "NAME_L3 <> 'Not Applicable'"); showGpMessage()
    gp.Dissolve_management(Output_Layer,Output_TargetGDB_CEC + "\\" +
Dissolved_FeatureClass_Root,"NAME_L3"); showGpMessage()
    gp.RepairGeometry_management(Output_TargetGDB_CEC + "\\" + Dissolved_FeatureClass_Root);
showGpMessage()
    gp.Delete_management(Output_Layer); showGpMessage()
    gp.Delete_management(Output_TargetGDB_CEC + "\\" + CECSource_FeatureClass_Root);
showGpMessage()
    print 'Created CEC AOA GDB and source feature class'

    # For CEC GDB, add and populate Feature Datasets for CONUS (includes PR and VI), Alaska and Hawaii
(marine only)

gp.CreateFeatureDataset_management(Output_TargetGDB_CEC,Conus_FeatureDataset,ConusSpatialRef);
showGpMessage()
    gp.CreateFeatureDataset_management(Output_TargetGDB_CEC,AK_FeatureDataset,AKSpatialRef);
showGpMessage()

gp.CreateFeatureDataset_management(Output_TargetGDB_CEC,Hawaii_FeatureDataset,HawaiiSpatialRef);
showGpMessage()

    # Get features from dissolved source
    gp.MakeFeatureLayer_management(Output_TargetGDB_CEC + "\\" + Dissolved_FeatureClass_Root,
Output_Select); showGpMessage()

    gp.SelectLayerByAttribute_management(Output_Select, "NEW_SELECTION", "NAME_L3 IN
("+Alaska_CECs+")"); showGpMessage()

```

```

gp.CopyFeatures_management(Output_Select, Output_TargetGDB_CEC + "\\" + AK_FeatureDataset + "\\"
+ AK_FeatureClass); showGpMessage()
gp.SelectLayerByAttribute_management(Output_Select, "CLEAR_SELECTION", ""); showGpMessage()
gp.RepairGeometry_management(Output_TargetGDB_CEC + "\\" + AK_FeatureDataset + "\\" +
AK_FeatureClass); showGpMessage()

gp.SelectLayerByAttribute_management(Output_Select, "NEW_SELECTION", "NAME_L3 IN
("+Hawaii_CECs+")"); showGpMessage()
gp.CopyFeatures_management(Output_Select, Output_TargetGDB_CEC + "\\" + Hawaii_FeatureDataset
+ "\\" + Hawaii_FeatureClass); showGpMessage()
gp.SelectLayerByAttribute_management(Output_Select, "CLEAR_SELECTION", ""); showGpMessage()
gp.RepairGeometry_management(Output_TargetGDB_CEC + "\\" + Hawaii_FeatureDataset + "\\" +
Hawaii_FeatureClass); showGpMessage()

gp.SelectLayerByAttribute_management(Output_Select, "NEW_SELECTION", "NAME_L3 IN
("+Marine_CECs+")"); showGpMessage()
gp.CopyFeatures_management(Output_Select, Output_TargetGDB_CEC +
"\\" + Conus_FeatureDataset + "\\" + Conus_FeatureClass); showGpMessage()
gp.SelectLayerByAttribute_management(Output_Select, "CLEAR_SELECTION", ""); showGpMessage()
gp.RepairGeometry_management(Output_TargetGDB_CEC + "\\" + Conus_FeatureDataset + "\\" +
Conus_FeatureClass); showGpMessage()

gp.workspace = Output_TargetGDB_CEC
fds = gp.ListDatasets("", "Feature"); showGpMessage()

for fd in fds:
    if fd == 'Alaska':
        cec_List = Alaska_CEC_Array
    elif fd == 'Hawaii':
        cec_List = Hawaii_CEC_Array
    else: cec_List = Marine_CEC_Array
    #else cec_List = [s for s in Conus_Parks.split()]
    gp.SelectLayerByAttribute(Output_Select, "CLEAR_SELECTION", ""); showGpMessage()

try:
    print "Extracting and buffering " + fd + " CECs"
    for cec in cec_List:
        buffered_FeatureClass = cec.replace(' ', '_') + "_" + Output_FeatureClass_Buffered_CEC
        cec_FeatureClass = cec.replace(' ', '_') + "_" + Output_FeatureClass_CEC
        gp.SelectLayerByAttribute_management(Output_Select, "NEW_SELECTION", "NAME_L3 =
\"" + cec + "\""); showGpMessage()
        gp.CopyFeatures_management(Output_Select, Output_TargetGDB_CEC + "\\" + fd + "\\" +
cec_FeatureClass); showGpMessage()
        gp.AddField_management(Output_TargetGDB_CEC + "\\" + fd + "\\" + cec_FeatureClass,
"AREA_SQKM", "FLOAT", "", "", "", "", "NULLABLE", "NON_REQUIRED", ""); showGpMessage()
        gp.CalculateField_management(Output_TargetGDB_CEC + "\\" + fd + "\\" + cec_FeatureClass,
"AREA_SQKM", "!shape.area! / 1000000", "PYTHON_9.3", ""); showGpMessage()
        gp.MetadataImporter(Metadata_Template, Output_TargetGDB_CEC + "\\" + fd + "\\" +
cec_FeatureClass); showGpMessage()
        gp.Buffer_analysis(Output_Select, Output_TargetGDB_CEC + "\\" + fd + "\\" +
buffered_FeatureClass, BufferDistance, "FULL", "ROUND", "NONE"); showGpMessage()
        gp.RepairGeometry_management(Output_TargetGDB_CEC + "\\" + fd + "\\" +
buffered_FeatureClass); showGpMessage()
        gp.AddField_management(Output_TargetGDB_CEC + "\\" + fd + "\\" + buffered_FeatureClass,
"AREA_SQKM", "FLOAT", "", "", "", "", "NULLABLE", "NON_REQUIRED", ""); showGpMessage()
        gp.CalculateField_management(Output_TargetGDB_CEC + "\\" + fd + "\\" +
buffered_FeatureClass, "AREA_SQKM", "!shape.area! / 1000000", "PYTHON_9.3", ""); showGpMessage()

```

```

        gp.SelectLayerByAttribute_management(Output_Select, "CLEAR_SELECTION", "");
showGpMessage()
        gp.MetadataImporter(Metadata_Template, Output_TargetGDB_CEC + "\\" + fd + "\\" +
cec_FeatureClass); showGpMessage()
        print "Completed " + cec
    except:
        print gp.getMessages()
        message = "\n\nERROR: Cannot buffer features in: " + cec_List
        raise Exception, message

    try:
        gp.workspace = fd
        merged_FeatureClass = fd + "_AOAs"
        fcList0 = ""
        fcs = gp.ListFeatureClasses("*_AOA_*", "POLYGON")
        for fc in fcs:
            fcList0 += Output_TargetGDB_CEC + "\\" + fd + "\\" + fc + ";";
        fcList = fcList0[0:-1] + ""
        gp.CreateFeatureClass(Output_TargetGDB_CEC + "\\" +
fd,merged_FeatureClass,"POLYGON",buffered_FeatureClass); showGpMessage()
        gp.Append_management(fcList,Output_TargetGDB_CEC + "\\" + fd + "\\" +
merged_FeatureClass,"TEST"); showGpMessage()
        gp.RepairGeometry_management(Output_TargetGDB_CEC + "\\" + fd + "\\" +
merged_FeatureClass); showGpMessage()
        #gp.JoinField_management(Output_TargetGDB_CEC + "\\" + fd + "\\" +
merged_FeatureClass,"NAME",Unit_Lookups,"NAME","NAME"); showGpMessage()
        print 'Created merged feature class ' + merged_FeatureClass
    except:
        print gp.getMessages()
        message = "\n\nERROR: Cannot create merged feature class for " + fd
        raise Exception, message

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

message = "Finished creating new NPScape CEC AOA geodatabase."
#del gp
print message

except:
    print 'PROCESSING ERROR:\n'
    message = "\n*** LAST GEOPROCESSOR MESSAGE (may not be source of the error)***";
showPyMessage()
showGpMessage()
message = "\n*** PYTHON ERRORS *** "; showPyMessage()
message = "Python Traceback Info: " + traceback.format_tb(sys.exc_info()[2])[0]; showPyMessage()
message = "Python Error Info: " + str(sys.exc_type)+ ": " + str(sys.exc_value) + "\n"; showPyMessage()
message = "\n*** PYTHON LOCAL VARIABLE LIST ***"; showPyMessage()

```

Create NewNPScape_LCC_AOAs.py

```
try:
    # Import system modules
    import sys, string, os, glob, datetime, arcgisscripting
    from datetime import datetime
    message = ""
    arcGISRoot = ""

    # Create the Geoprocessor object
    gp = arcgisscripting.create(9.3)

    gp.OverwriteOutput = 1

    # Load required toolboxes...
    if gp.Exists("C:/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx"):
        gp.AddToolbox("C:/arcgis/ArcToolbox/Toolboxes/Spatial Analyst Tools.tbx")
        gp.AddToolbox("C:/arcgis/ArcToolbox/Toolboxes/Conversion Tools.tbx")
        gp.AddToolbox("C:/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx")
        arcGISRoot = "C:\\arcgis"
    elif gp.Exists("C:/Program Files/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx"):
        gp.AddToolbox("C:/Program Files/arcgis/ArcToolbox/Toolboxes/Spatial Analyst Tools.tbx")
        gp.AddToolbox("C:/Program Files/arcgis/ArcToolbox/Toolboxes/Conversion Tools.tbx")
        gp.AddToolbox("C:/Program Files/arcgis/ArcToolbox/Toolboxes/Data Management Tools.tbx")
        arcGISRoot = "C:\\Program Files\\arcgis"
    else:
        message = "ERROR: Cannot find required ArcToolboxes in standard locations."
        gp.AddMessage(message) # This does absolutely nothing!
        raise Exception, message

    # Local variables...
    today=datetime.now()
    datestamp = str(today.isoformat()).replace('-',')[0:8]
    print datestamp

    NPScape_Workspace = "F:\\NPScape\\Scratch"

    FeatureClass_Root = "FWS_LCC_Albers"

    LCCSource_Folder = "O:\\SourceData\\NorthAmericaData\\Boundary"
    LCCSource_FeatureClass_Root = "FWS_LCC"
    LCCOutput_FeatureClass = "FWS_LCC_Albers"
    LCCSource_FeatureClass = LCCSource_Folder + "\\ " + LCCSource_FeatureClass_Root + ".shp"
    MasterSpatialRef = arcGISRoot + "\\Coordinate Systems\\Projected Coordinate Systems\\Continental\\North America\\USA Contiguous Albers Equal Area Conic USGS.prj"
    MasterTransform = "NAD_1983_To_WGS_1984_4"
    PacificTransform = "NAD_1983_To_WGS_1984_3"

    Output_TargetRoot = "NPScape_AOA"
    Output_TargetGDB_LCC = NPScape_Workspace + "\\ " + Output_TargetRoot + "_LCC.gdb"
    Versioned_Output_TargetGDB = NPScape_Workspace + "\\ " + Output_TargetRoot + "_LCC_" + datestamp + ".gdb"

    Output_FeatureClass_LCC = "AOA_LCC"
    Output_FeatureClass_Buffered_LCC = "AOA_LCC_10km"

    BufferDistance = "10 Kilometers"
```

```

#Unit_Lookups = "O:\\SourceData\\Lookups\\NPSCape_Unit_Info.mdb\\tlu_IM_Units"

AKSRSource = "O:\\SourceData\\USData\\Boundary\\AK_NPS_Boundary.gdb\\AK_Albers"
AKSRDesc = gp.Describe(AKSRSource)
AKSpatialRef = AKSRDesc.SpatialReference
PacificSRSource = "O:\\SourceData\\USData\\Boundary\\Pacific_Templates\\LCC_Pacific.shp"
PacificSRDesc = gp.Describe(PacificSRSource)
PacificSpatialRef = PacificSRDesc.SpatialReference
#PacificSpatialRef = arcGISRoot + "\\Coordinate Systems\\Projected Coordinate
Systems\\World\\Mercator (world).prj"
#PacificTransform = ""
# TODO: Use custom PacP00 transform/custom projection to change to NAD83 datum:
WGS_1984_To_NAD_1983_PACP00.gtf
AK_FeatureDataset = "Alaska"
Pacific_FeatureDataset = "Pacific"
Conus_FeatureDataset = "CONUS"
AK_FeatureClass = "LCC_Alaska"
Pacific_FeatureClass = "LCC_Pacific"
Pacific_FeatureClass_Temp = "LCC_Pacific_Temp"
Conus_FeatureClass = "LCC_CONUS"

Alaska_LCCs = "Arctic','North Pacific','Aleutian and Bering Sea Islands','Northwestern Interior
Forest','Western Alaska'"
Alaska_LCC_Array = ['Arctic','North Pacific','Aleutian and Bering Sea Islands','Northwestern Interior
Forest','Western Alaska']
#Alaska_LCC_Array = [11,17,18,19,20]
Pacific_LCCs = "'Pacific Islands'"
Pacific_LCC_Array = ['Pacific Islands']
#Pacific_LCC_Array = [21]
Conus_LCCs = "'Appalachian','California','Desert','Eastern Tallgrass Prairie and Big Rivers','Great
Basin','Great Northern','Great Plains','Gulf Coast Prairie','Gulf Coastal Plains and Ozarks','North
Atlantic','Peninsular Florida','Plains and Prairie Potholes','South Atlantic','Southern Rockies','Upper Midwest
and Great Lakes'"
Conus_LCC_Array = ['Appalachian','California','Desert','Eastern Tallgrass Prairie and Big Rivers','Great
Basin','Great Northern','Great Plains','Gulf Coast Prairie','Gulf Coastal Plains and Ozarks','North
Atlantic','Peninsular Florida','Plains and Prairie Potholes','South Atlantic','Southern Rockies','Upper Midwest
and Great Lakes']

Output_Layer = "AOABounds_Layer"
Output_Select = "AOABounds_Select"
LCC_Layer = "LCC_Layer"

OutputsToDelete = [Versioned_Output_TargetGDB, Output_Layer, Output_Select, LCC_Layer]

for output in OutputsToDelete:
    if gp.Exists(output):
        gp.Delete(output)

try:
    if gp.Exists(Output_TargetGDB_LCC):
        gp.Copy(Output_TargetGDB_LCC, Versioned_Output_TargetGDB)
        gp.Delete(Output_TargetGDB_LCC)

gp.CreateFileGDB(NPSCape_Workspace, Output_TargetRoot + "_LCC.gdb")

# Copy LCC feature class into new file geodatabases
gp.MakeFeatureLayer(LCCSource_FeatureClass,LCC_Layer)

```

```

gp.Project_management(LCC_Layer, Output_TargetGDB_LCC + "\\\" + LCCOutput_FeatureClass,
MasterSpatialRef, MasterTransform)
gp.Delete(LCC_Layer)

gp.RepairGeometry(Output_TargetGDB_LCC + "\\\" + LCCOutput_FeatureClass)

ConusSRSource = Output_TargetGDB_LCC + "\\\" + LCCOutput_FeatureClass
ConusSRDesc = gp.Describe(ConusSRSource)
ConusSpatialRef = ConusSRDesc.SpatialReference

# For LCC GDB, add and populate Feature Datasets for CONUS (includes PR and VI), Alaska and Pacific
Islands
gp.CreateFeatureDataset(Output_TargetGDB_LCC,Conus_FeatureDataset,ConusSpatialRef)
gp.CreateFeatureDataset(Output_TargetGDB_LCC,AK_FeatureDataset,AKSpatialRef)
gp.CreateFeatureDataset(Output_TargetGDB_LCC,Pacific_FeatureDataset,PacificSpatialRef)

gp.MakeFeatureLayer(Output_TargetGDB_LCC + "\\\" + LCCOutput_FeatureClass, Output_Select)

gp.SelectLayerByAttribute(Output_Select, "NEW_SELECTION", "area_names IN (" + Alaska_LCCs + ")")
gp.CopyFeatures(Output_Select, Output_TargetGDB_LCC + "\\\" + AK_FeatureDataset + "\\\" +
AK_FeatureClass)
gp.SelectLayerByAttribute(Output_Select, "CLEAR_SELECTION", "")
gp.RepairGeometry(Output_TargetGDB_LCC + "\\\" + AK_FeatureDataset + "\\\" + AK_FeatureClass)

gp.SelectLayerByAttribute(Output_Select, "NEW_SELECTION", "area_names IN (" + Pacific_LCCs + ")")
gp.CopyFeatures(Output_Select, Output_TargetGDB_LCC + "\\\" + Pacific_FeatureDataset + "\\\" +
Pacific_FeatureClass)
gp.SelectLayerByAttribute(Output_Select, "CLEAR_SELECTION", "")
gp.RepairGeometry(Output_TargetGDB_LCC + "\\\" + Pacific_FeatureDataset + "\\\" + Pacific_FeatureClass)

gp.SelectLayerByAttribute(Output_Select, "NEW_SELECTION", "area_names IN (" + Alaska_LCCs + ")")
gp.SelectLayerByAttribute(Output_Select, "ADD_TO_SELECTION", "area_names IN (" + Pacific_LCCs + ")")
gp.SelectLayerByAttribute(Output_Select, "SWITCH_SELECTION")
gp.CopyFeatures(Output_Select, Output_TargetGDB_LCC + "\\\" + Conus_FeatureDataset + "\\\" +
Conus_FeatureClass)
gp.SelectLayerByAttribute(Output_Select, "CLEAR_SELECTION", "")
gp.RepairGeometry(Output_TargetGDB_LCC + "\\\" + Conus_FeatureDataset + "\\\" + Conus_FeatureClass)

# For each feature class by LCC_Number, select feature, copy to Output_TargetGDB_LCC, then buffer to
10km
gp.workspace = Output_TargetGDB_LCC
fds = gp.ListDatasets("", "Feature")

for fd in fds:
    if fd == 'Alaska':
        lcc_List = Alaska_LCC_Array
    elif fd == 'Pacific':
        lcc_List = Pacific_LCC_Array
    else: lcc_List = Conus_LCC_Array
    #else lcc_List = [s for s in Conus_Parks.split()]

    try:
        print "Extracting and buffering " + fd + " LCCs"
        for lcc in lcc_List:
            buffered_FeatureClass = lcc.replace(' ', '_') + "_" + Output_FeatureClass_Buffered_LCC
            lcc_FeatureClass = lcc.replace(' ', '_') + "_" + Output_FeatureClass_LCC
            gp.SelectLayerByAttribute(Output_Select, "NEW_SELECTION", "area_names = \" + lcc + "\"")
            gp.CopyFeatures(Output_Select, Output_TargetGDB_LCC + "\\\" + fd + "\\\" + lcc_FeatureClass)

```

```

        gp.DeleteField(Output_TargetGDB_LCC + "\\\" + fd + "\\\" + lcc_FeatureClass, "Shape_Leng")
        gp.AddField_management(Output_TargetGDB_LCC + "\\\" + fd + "\\\" + lcc_FeatureClass,
"AREA_SQKM", "FLOAT", "11", "3", "", "", "NULLABLE", "NON_REQUIRED", "")
        gp.CalculateField_management(Output_TargetGDB_LCC + "\\\" + fd + "\\\" + lcc_FeatureClass,
"AREA_SQKM", "!shape.area! / 1000000", "PYTHON_9.3", "")
        gp.Buffer_analysis(Output_Select, Output_TargetGDB_LCC + "\\\" + fd + "\\\" +
buffered_FeatureClass, BufferDistance, "FULL", "ROUND", "NONE")
        gp.RepairGeometry(Output_TargetGDB_LCC + "\\\" + fd + "\\\" + buffered_FeatureClass)
        gp.AddField_management(Output_TargetGDB_LCC + "\\\" + fd + "\\\" + buffered_FeatureClass,
"AREA_SQKM", "FLOAT", "11", "3", "", "", "NULLABLE", "NON_REQUIRED", "")
        gp.CalculateField_management(Output_TargetGDB_LCC + "\\\" + fd + "\\\" +
buffered_FeatureClass, "AREA_SQKM", "!shape.area! / 1000000", "PYTHON_9.3", "")
        gp.SelectLayerByAttribute(Output_Select, "CLEAR_SELECTION", "")
        print "Completed " + lcc

    except:
        print gp.getMessages()
        message = "\n\nERROR: Cannot buffer features in: " + lcc_List
        raise Exception, message

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

message = "Finished creating new NPScape LCC AOA geodatabases."
#del gp
print message
except:
    error = gp.GetMessages() + ' ' + message
    #del gp
    print error

```