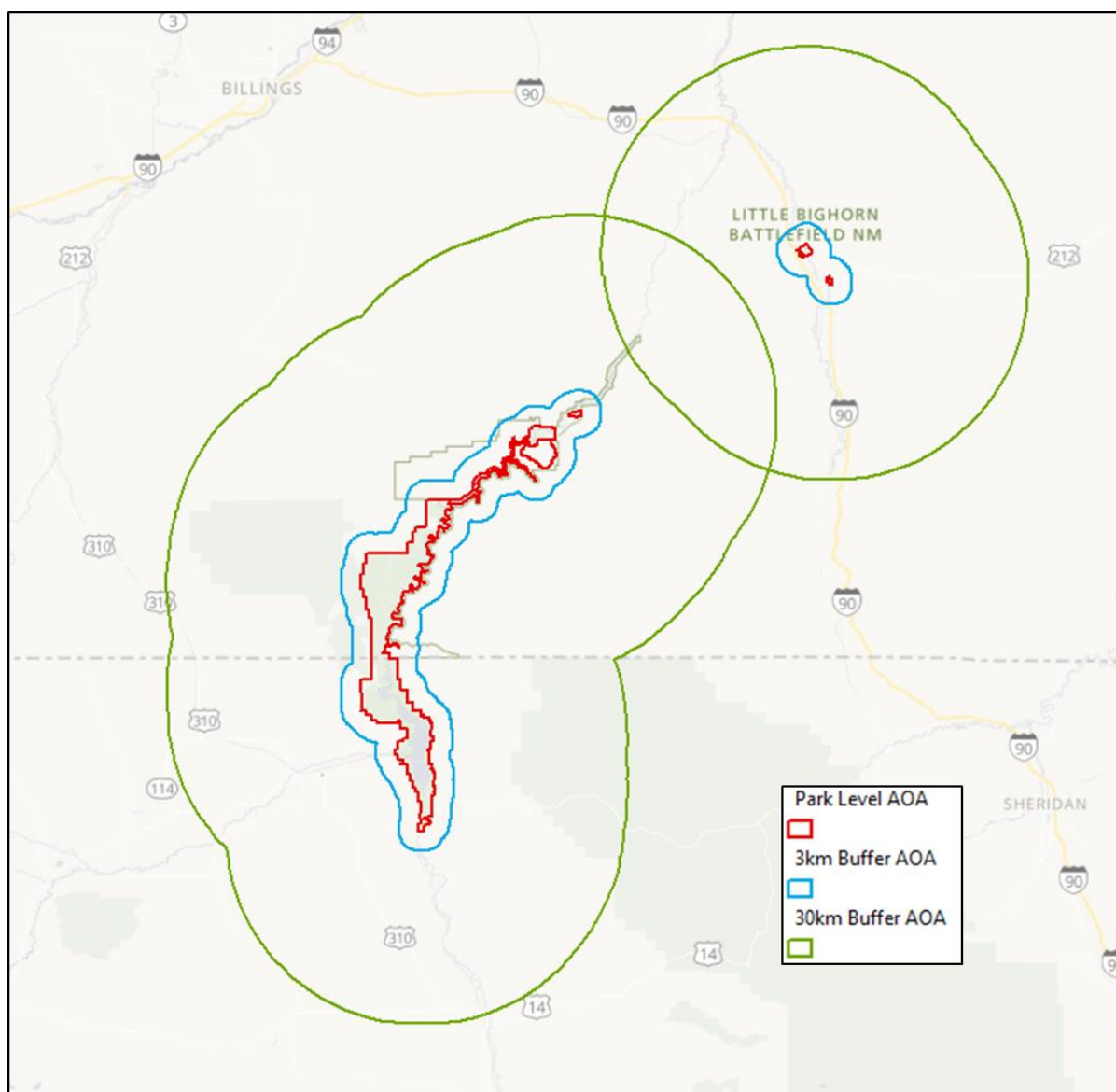




Standard Areas of Analysis for use in the National Park Environmental Settings Monitoring Protocol

2018

Natural Resource Data Series NPS/NRSS/NRDS—2020/1271



ON THE COVER

Map of the standard areas of analysis (AOAs) for Bighorn Canyon National Recreation Area and Little Bighorn Battlefield National Monument
(NPS Inventory and Monitoring Division)

Standard Areas of Analysis for use in the National Park Environmental Settings Monitoring Protocol 2018

Natural Resource Data Series NPS/NRSS/NRDS—2020/1271

Michelle Kinseth,¹ Lisa Nelson,²

¹ Managed Business Solutions (MBS), a Sealaska Company
Contractor to National Park Service
Natural Resource Stewardship and Science
Inventory & Monitoring Division
1201 Oakridge Dr., Suite 150
Fort Collins, CO 80525

² National Park Service
Natural Resource Stewardship and Science
Inventory and Monitoring Division
1201 Oakridge Dr., Suite 150
Fort Collins, CO 80525

March 2020

U.S. Department of the Interior
National Park Service
Natural Resource Stewardship and Science
Fort Collins, Colorado

The National Park Service, Natural Resource Stewardship and Science office in Fort Collins, Colorado, publishes a range of reports that address natural resource topics. These reports are of interest and applicability to a broad audience in the National Park Service and others in natural resource management, including scientists, conservation and environmental constituencies, and the public.

The Natural Resource Data Series is intended for the timely release of basic data sets and data summaries. Care has been taken to assure accuracy of raw data values, but a thorough analysis and interpretation of the data has not been completed. Consequently, the initial analyses of data in this report are provisional and subject to change.

All manuscripts in the series receive the appropriate level of peer review to ensure that the information is scientifically credible, technically accurate, appropriately written for the intended audience, and designed and published in a professional manner.

The creation of data sets documented in this report followed peer reviewed methods described or referenced herein. Resultant data sets were reviewed to ensure accuracy, completeness, and consistency with documented data quality standards, as well as for usability and reproducibility.

Views, statements, findings, conclusions, recommendations, and data in this report do not necessarily reflect views and policies of the National Park Service, U.S. Department of the Interior. Mention of trade names or commercial products does not constitute endorsement or recommendation for use by the U.S. Government.

This report is available in digital format from the [Natural Resource Publications Management website](#). To receive this report in a format that is optimized to be accessible using screen readers for the visually or cognitively impaired, please email irma@nps.gov.

Please cite this publication as:

Kineth, M. and L. Nelson. 2020. Standard areas of analysis for use in the national park environmental settings monitoring protocol: 2018. Natural Resource Data Series NPS/NRSS/NRDS—2020/1271. National Park Service, Fort Collins, Colorado.

Contents

	Page
Figures.....	iv
Tables.....	v
Appendices.....	vi
Abstract.....	vii
Background and Summary.....	1
Methods.....	2
Source Data	5
Alternate Boundaries.....	5
I&M Operational Boundaries.....	5
Creation of Interim Landscape Dynamics Boundaries.....	6
Inactivation of Obsolete Landscape Dynamics Boundaries.....	7
Creation of Park, 3km, and 30km Standard AOAs	7
Inactivation of Obsolete AOAs	10
Additional Processing.....	10
Code availability.....	11
Data Records.....	12
Data Quality Evaluation.....	16
Usage Notes	16
References.....	17

Figures

	Page
Figure 1. General process flow for the creation of standard landscape dynamics AOAs.....	2
Figure 2. Data structure for the Hawaii geography within the landscape dynamics standard AOA database, with AOAs organized by park, 3km, and 30km feature datasets (NPS IMD 2018c).	8
Figure 3. Data structure for the landscape dynamics standard combined AOAs (service-wide) database, with AOAs organized by park, 3km, and 30km feature classes (NPS IMD 2018d).	9
Figure A-1. Spatial extent of park AOAs (in green) in the Continental United States.	19
Figure A-2. Spatial extent of park AOAs in Alaska (in green).	20
Figure A-3. Spatial extent of park AOAs in Hawaii (in green).	20
Figure A-4. Spatial extent of park AOAs in American Samoa (in green).	21
Figure A-5. Spatial extent of park AOAs in U.S. Virgin Islands (in green).	21
Figure A-6. Spatial extent of park AOAs (in green) in Saipan (left) and Guam (right).	22

Tables

	Page
Table 1. Updated IMD operational boundaries leading to an update in standard landscape dynamics standard.....	3
Table 2. Status and count of landscape dynamics alternate boundary spatial features in the alternate boundaries source dataset (NPS IMD 2018a).	6
Table 3. Feature alteration codes and descriptions for interim landscape dynamics boundaries when compared to source dataset IMD operational boundaries.....	6
Table 4. Feature alteration attribute counts in the interim landscape dynamics boundaries when compared to input IMD operational boundaries source.	7
Table 5. Features with a change to lifecycle status in the interim landscape dynamics boundaries dataset.	7
Table 6. Active and inactive feature counts in the interim landscape dynamics boundaries dataset.	7
Table 7. Standard AOA feature counts, by geography.....	9
Table 8. Counts of change reason from preceding version for standard AOA datasets by geography, and service-wide.....	9
Table 9. Active and inactive feature counts for standard AOAs datasets by geography, and service-wide.	10
Table 10. Data sets and their provenance. Temporal currency of datasets is based on the date of publication of Land Resources Division official boundaries.....	15

Appendices

	Page
Appendix A. Park AOA Geographies.....	19
Appendix B. Python Toolbox Code to Create Landscape Dynamics Bounds.....	23
Appendix C. Python Toolbox Code to Create AOAs	32
Appendix D. Additional Processing Script for Standard AOAs	52
Appendix E. Session and Version Information.....	64

Abstract

This data report (and associated datasets) documents the acquisition, preparation, and creation of National Park Service (NPS) standard areas of analysis (AOAs) for use within the Inventory and Monitoring Program (I&M Program). AOA datasets were updated through September 30, 2018 and supersede all previous releases/versions.

The Inventory and Monitoring Division, under the I&M Program, works with a select subset of the more than 400 park units of the National Park System; NPS parks that are part of the I&M Program are included in this subset. Landscape dynamics, weather, and stream hydrology measures are monitored using methods outlined in the National Park Environmental Settings Monitoring Protocol (DeVivo et.al 2018), which includes areas inside and near parks with significant natural resources. Annually, standard AOAs are updated or created based on official boundary changes and/or requests made by parks or monitoring networks throughout the year. AOA datasets documented in this report may not include all NPS units, or other NPS areas. These AOAs are not official boundaries and are not intended for use outside NPS Inventory and Monitoring.

Background and Summary

The status and condition of park resources is affected by interconnected system drivers at regional to global scales surrounding parks. Landscape dynamics provide an important context for understanding patterns and changes in resources monitored at parks. The NPS Environmental Settings Monitoring Protocol produces nationally-standardized status and trend information related to *Landscape Dynamics, Climate, and Surface Water Dynamics* metrics (DeVivo et al. 2018).

Annually, IMD calculates protocol metrics pertaining to landscape dynamics, hydrology, and climate for areas of analysis (AOAs). AOAs are defined areas of interest. Standard AOAs include three local extents: the park boundary, a 3km buffer of the park boundary, and a 30km buffer of the park boundary (see report cover).

Standard AOAs are created annually from IMD operational boundaries. In some cases, an AOA park level boundary may differ from the source IMD operational boundary, by park or network request. In these instances, for the sole purposes of creating AOAs for use with the protocol, a boundary that may better represent management and planning work at a park is provided by a park or monitoring network. This park or network-provided boundary replaces the IMD operational boundary in output AOA datasets.

Methods

This report describes the creation of standard landscape dynamics AOAs which are derived from the IMD operational boundaries and alternate boundaries datasets maintained by IMD staff (Kinseth and Nelson 2019). Not all IMD operational boundaries are landscape dynamics boundaries. Of the 293 active features in IMD operational boundaries, 281 become active features in the interim landscape dynamics boundaries dataset. These interim landscape dynamics boundaries generate landscape dynamics standard AOAs (Figure 1) for use in the National Park Environmental Settings Monitoring Protocol.

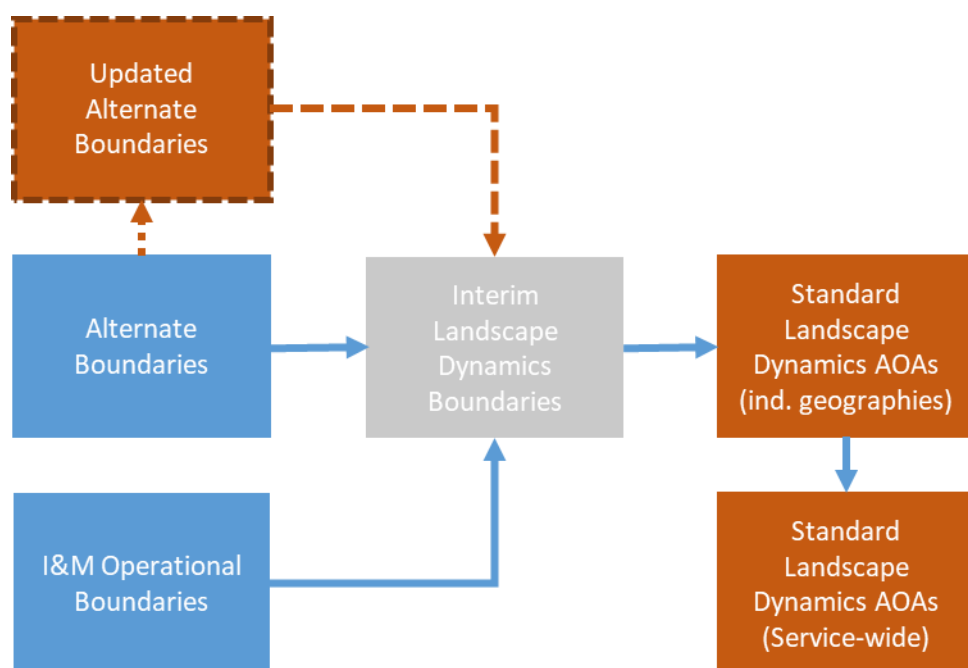


Figure 1. General process flow for the creation of standard landscape dynamics AOAs. Orange items indicate datasets described and published in this report; dashed lines indicate a portion of the process and dataset generation that may not be generated annually. Gray items are interim datasets and are not published.

This report also documents the incorporation of all IMD operational boundary updates that occurred from September 30, 2017 through September 30, 2018 into standard landscape dynamics AOAs (NPS IMD 2018c, NPS IMD 2018d). In that time period, 38 IMD operational boundaries were updated (Table 1), triggering an update of standard AOAs for the same units.

Table 1. Updated IMD operational boundaries leading to an update in standard landscape dynamics standard AOAs. IMD Create Date is the date the AOA was created from the input source dataset and might not coincide with the Land Resources Division (LRD) boundary release date (Edit Date).

Unit Name	Unit Code	Edit Date (by LRD)	IMD Create Date	Change Reason from Preceding Version
Grant-Kohrs Ranch National Historic Site	GRKO	11/30/2017	1/19/2018	Changed by NPS LRD
Chaco Culture National Historical Park	CHCU	12/11/2017	1/19/2018	Changed by NPS LRD
Homestead National Monument of America	HOME	12/14/2017	1/19/2018	Changed by NPS LRD
Petrified Forest National Park	PEFO	1/9/2018	4/4/2018	Changed by NPS LRD
Big Bend National Park	BIBE	1/9/2018	4/4/2018	Changed by NPS LRD
Katahdin Woods and Waters National Monument	KAWW	1/23/2018	4/4/2018	Changed by NPS LRD
Washita Battlefield National Historic Site	WABA	1/24/2018	4/4/2018	Changed by NPS LRD
Wupatki National Monument	WUPA	2/6/2018	4/4/2018	Changed by NPS LRD
Walnut Canyon National Monument	WACA	2/12/2018	4/4/2018	Changed by NPS LRD
Black Canyon of the Gunnison National Park	BLCA	2/15/2018	4/4/2018	Changed by NPS LRD
Minute Man National Historical Park	MIMA	2/22/2018	4/4/2018	Changed by NPS LRD
Hawaii Volcanoes National Park	HAVO	2/26/2018	2/26/2018	Changed by NPS LRD
San Antonio Missions National Historical Park	SAAN	2/28/2018	4/4/2018	Changed by NPS LRD
Saint Croix National Scenic River	SACN	3/14/2018	4/4/2018	Changed by NPS LRD
Rio Grande Wild and Scenic River	RIGR	3/22/2018	4/4/2018	Changed by NPS LRD
Big Thicket National Preserve	BITH	3/27/2018	4/4/2018	Changed by NPS LRD
Great Sand Dunes National Park and Preserve	GRSA	3/28/2018	4/4/2018	Changed by NPS LRD

Table 1 (continued). Updated IMD operational boundaries leading to an update in standard landscape dynamics standard AOAs. IMD Create Date is the date the AOA was created from the input source dataset and might not coincide with the Land Resources Division (LRD) boundary release date (Edit Date).

Unit Name	Unit Code	Edit Date (by LRD)	IMD Create Date	Change Reason from Preceding Version
Stones River National Battlefield	STRI	3/29/2018	4/4/2018	Changed by NPS LRD
Chickasaw National Recreation Area	CHIC	3/29/2018	4/4/2018	Changed by NPS LRD
Tumacacori National Historical Park	TUMA	4/5/2018	7/16/2018	Changed by NPS LRD
Fort Laramie National Historic Site	FOLA	4/9/2018	7/16/2018	Changed by NPS LRD
Arches National Park	ARCH	4/9/2018	7/16/2018	Changed by NPS LRD
Fort Davis National Historic Site	FODA	4/27/2018	7/16/2018	Changed by NPS LRD
Cedar Breaks National Monument	CEBR	5/4/2018	7/16/2018	Changed by NPS LRD
Mississippi National River and Recreation Areas	MISS	5/10/2018	7/16/2018	Changed by NPS LRD
Valley Forge National Historical Park	VAFO	5/14/2018	7/16/2018	Changed by NPS LRD
Delaware Water Gap National Recreation Area	DEWA	5/25/2018	7/16/2018	Changed by NPS LRD
Colonial National Historical Park	COLO	6/11/2018	7/16/2018	Changed by NPS LRD
Rocky Mountain National Park	ROMO	6/18/2018	7/16/2018	Changed by NPS LRD
Saratoga National Historical Park	SARA	7/2/2018	10/3/2018	Changed by NPS LRD
Mesa Verde National Park	MEVE	7/18/2018	10/3/2018	Changed by NPS LRD
Fort Union Trading Post National Historic Site	FOUS	7/24/2018	10/3/2018	Changed by NPS LRD
Grand Canyon National Park	GRCA	7/25/2018	10/3/2018	Changed by NPS LRD
Padre Island National Seashore	PAIS	7/26/2018	10/3/2018	Changed by NPS LRD

Table 1 (continued). Updated IMD operational boundaries leading to an update in standard landscape dynamics standard AOAs. IMD Create Date is the date the AOA was created from the input source dataset and might not coincide with the Land Resources Division (LRD) boundary release date (Edit Date).

Unit Name	Unit Code	Edit Date (by LRD)	IMD Create Date	Change Reason from Preceding Version
Hot Springs National Park	HOSP	8/23/2018	10/3/2018	Changed by NPS LRD
Petroglyph National Monument	PETR	8/24/2018	10/3/2018	Changed by NPS LRD
Tule Springs Fossil Beds National Monument	TUSK	9/26/2018	10/3/2018	Changed by NPS LRD
Indiana Dunes National Lakeshore	INDU	9/27/2018	10/3/2018	Changed by NPS LRD

Source Data

Generating standard landscape dynamics AOAs requires two source datasets, alternate boundaries (NPS IMD 2018a) and IMD operational boundaries (NPS IMD 2018b).

Alternate Boundaries

Alternate Boundaries may be altered or updated annually by IMD staff (NPS IMD 2018a). The dataset includes boundaries intended for use specifically in environmental setting protocol metric calculation, and these boundaries are used to replace existing boundaries.

I&M Operational Boundaries

I&M Operational Boundaries are created up to quarterly by IMD staff (NPS IMD 2018b). The IMD operational boundaries dataset is derived from the official administrative boundary quarterly update by the NPS Land Resources Division, or LRD (NPS LRD 2018), as described in Kinseth and Nelson, 2020.

Incorporation of Boundary Changes from Parks and Networks

If, in the context of the environmental setting protocol, a monitoring network or park has an alternate boundary they wish to use, replacement of the default LRD boundary occurs during creation of the interim landscape dynamics boundaries dataset. The replacement of a park boundary with an alternate version is by direct request from a monitoring network or park. For the period documented in this report, one landscape dynamics alternate boundary was incorporated; the Greater Yellowstone Network provided an alternate park boundary for Grand Teton National Park (GRTE).

Landscape dynamics parks in the alternate boundaries dataset are removed or inactivated if one or more of these criteria are met: (1) a park or monitoring network submits a new alternate boundary; or (2) a park or monitoring network rescinds the request to use an existing alternate boundary; or (3) a park or monitoring network submits an update of an existing alternate boundary. No new or amended alternate boundaries were received from park or network staff for the time period documented in this

data report. Only active alternate boundary features (GRTE, as noted above) are used for creation of interim landscape dynamics boundaries and subsequent standard AOAs (Table 2).

Table 2. Status and count of landscape dynamics alternate boundary spatial features in the alternate boundaries source dataset (NPS IMD 2018a).

Source Dataset	Active	Inactive	Total Features
Alternate Boundary – Landscape Dynamics	1	0	1

Creation of Interim Landscape Dynamics Boundaries

The interim landscape dynamics boundary dataset is a projected polygon dataset representing landscape dynamics boundaries (IMD_LD_BND_albers). This interim landscape dynamics boundaries dataset is used to create landscape dynamics standard AOAs (NPS IMD 2018c).

To create the interim landscape dynamics boundaries dataset, a subset of active features from the IMD operational boundaries dataset IMD_BND_Units_py (NPS IMD 2018b) was determined by cross-referencing the dataset with the landscape dynamics park list (NPS IRMA 2018). This subset of park boundaries was combined with the landscape dynamics alternate boundary dataset (NPS IMD 2018a) to create the interim landscape dynamics boundaries dataset (Kinseth and Nelson 2018a). Replacement or addition of features in the creation of the interim landscape dynamics boundaries dataset takes place annually and is tracked using codes in the feature alteration attribute field (Table 3).

Table 3. Feature alteration codes and descriptions for interim landscape dynamics boundaries when compared to source dataset IMD operational boundaries.

Code	Description
NC	No alteration to output feature from input source (Default value)
AF	Added feature; feature was not present in input source
RS	Replaced feature; input source feature inappropriate for intended use
MS	Input source features merged to create one distinct output feature
SS	Input source feature split to create two or more distinct output features

The feature alteration attribute value identifies how, if at all, a feature in the interim landscape dynamics boundary dataset (and subsequent standard AOAs) differs from the respective feature in the IMD operational boundary source dataset. For the time period covered by this report, one IMD operational boundary feature was replaced by an alternate boundary (Table 4).

Table 4. Feature alteration attribute counts in the interim landscape dynamics boundaries when compared to input IMD operational boundaries source.

Category	No Alteration to Feature (NC)	Added Feature (AF)	Replaced Feature (RS)	Merged Feature (MS)	Split Feature (SS)	Total Features
Landscape Dynamics Boundaries	281	0	1	0	0	282

Inactivation of Obsolete Landscape Dynamics Boundaries

Only active features in the interim landscape dynamics boundaries dataset are used to create standard AOAs. One feature in the interim landscape dynamics boundaries dataset was inactivated and replaced with a new feature. Attribute values for *Geometry ID* (unique identifier) and *Unit Code Display* reflect the update (Table 5). The feature being replaced was assigned a lifecycle status of ‘inactive’ (Tables 5 and 6) and the new feature was assigned a lifecycle status of ‘active’.

Table 5. Features with a change to lifecycle status in the interim landscape dynamics boundaries dataset.

Unit Name	Unit Code	Unit Code Display	Geometry ID	Status
Grand Teton National Park	GRTE	GRTE	IMD-Park-102039-GRTE-20180404	Inactive
Grand Teton National Park	GRTE	GRTE_JODR	IMD-Park-102039-GRTE_JODR-20161007	Active

Table 6. Active and inactive feature counts in the interim landscape dynamics boundaries dataset.

Status	Count
Active	281
Inactive	1
Total	282

Creation of Park, 3km, and 30km Standard AOAs

Standard AOAs are defined at local extents: park-level, 3km-buffer of the park, and 30km-buffer of the park. One or more standard AOAs are then collected into feature datasets by geography (Alaska, CONUS, Hawaii, Guam, American Samoa, Commonwealth Northern Mariana Islands, and Puerto Rico and U.S. Virgin Islands), primarily defined by the use of the same spatial reference. All geographies are combined to create service-wide feature classes.

Active features from the interim landscape dynamics boundaries dataset were grouped by geography and buffered (Kinseth and Nelson 2018b) to create feature datasets in the output database Landscape Dynamics Standard AOAs (NPS IMD 2018c). The local extent AOAs were all combined to create service-wide AOA feature classes in the output database Combination Landscape Dynamics Standard AOAs (NPS IMD 2018d). Except where explicitly stated, all subsequent references to landscape dynamics standard AOAs includes both AOA databases.

As shown in the example in Figure 2, within each geography are features for each park found in that geographic area, as well as a combined dataset of all parks (by AOA type): park level AOAs (park boundary), 3km AOAs (park boundary buffered 3km), and 30km AOAs (park boundary buffered 30km). As noted above, geographies include: Alaska, American Samoa, Commonwealth of Northern Mariana Islands (Saipan), Continental U.S. (CONUS), Hawaii, Guam, and U.S. Virgin Islands (USVI) (Table 7). See Appendix A for maps of the various geographies. There are no IMD operational boundaries and therefore no standard AOAs in Puerto Rico.

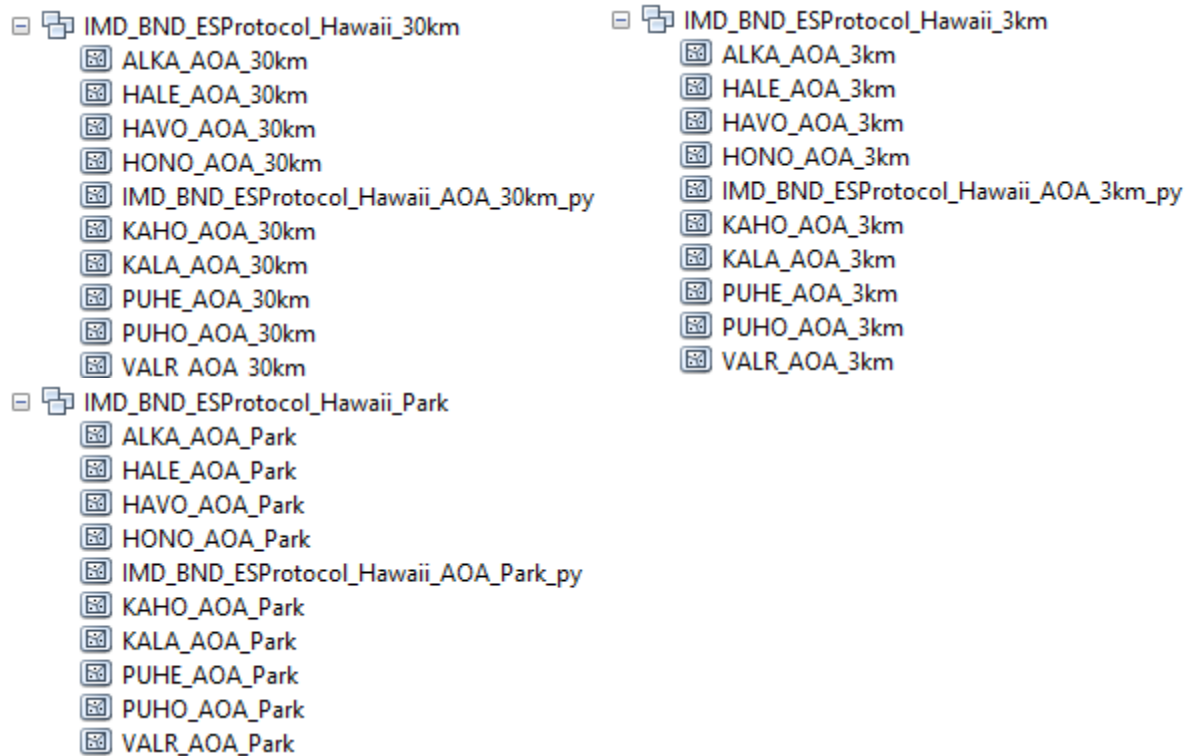


Figure 2. Data structure for the Hawaii geography within the landscape dynamics standard AOA database, with AOAs organized by park, 3km, and 30km feature datasets (NPS IMD 2018c).

In the landscape dynamics standard combined AOAs output database are feature classes for each AOA type (park, 3km, 30km) that include all landscape dynamics parks (281) service-wide. As shown in Figure 3, service-wide AOAs are contained in combined feature classes, organized by park, 3km, and 30km. Two different spatial references are available, WGS 1984 (WGS), and USA Contiguous Albers Equal Area Conic USGS version (Albers).

 IMD_BND_ESProtocol_30km_albers_py
 IMD_BND_ESProtocol_30km_wgs_py
 IMD_BND_ESProtocol_3km_albers_py
 IMD_BND_ESProtocol_3km_wgs_py
 IMD_BND_ESProtocol_Park_albers_py
 IMD_BND_ESProtocol_Park_wgs_py

Figure 3. Data structure for the landscape dynamics standard combined AOAs (service-wide) database, with AOAs organized by park, 3km, and 30km feature classes (NPS IMD 2018d).

Table 7. Standard AOA feature counts, by geography. Feature count is the same across all AOA types (park, 3km, 30km) for each geography, and service-wide.

Geography	AOA Feature Count
Alaska	16
American Samoa	1
Saipan (CNMI)	1
CONUS	251
Hawaii	9
Guam	1
USVI	2
Combined (service-wide)	281

The “change reason from preceding version” attribute, in the landscape dynamics standard AOAs datasets, was inherited from the landscape dynamics boundary interim dataset (Table 8). The “change reason” only applies to the current version of landscape dynamics standard AOAs (NPS IMD 2018c, NPS IMD 2018d) when compared to the preceding year’s version of landscape dynamics standard AOAs (NPS IMD 2017).

Table 8. Counts of change reason from preceding version for standard AOA datasets by geography, and service-wide. Count is the same across all AOA types (park, 3km, 30km).

Geography	Change Reason from Preceding Version		
	Changed by NPS LRD	No Change	Count
Alaska	0	16	16
American Samoa	0	1	1
Saipan (CNMI)	0	1	1

Table 8 (continued). Counts of change reason from preceding version for standard AOA datasets by geography, and service-wide. Count is the same across all AOA types (park, 3km, 30km).

Geography	Change Reason from Preceding Version		Count
	Changed by NPS LRD	No Change	
CONUS	37	214	251
Hawaii	1	8	9
Guam	0	1	1
U.S. Virgin Islands	0	2	2
Combined (service-wide)	38	243	281

Inactivation of Obsolete AOAs

In the landscape dynamics standard AOAs database, any previously existing AOA features with an update, or features to be retired, are assigned a lifecycle status of ‘inactive’ (Table 9). No obsolete AOA features were flagged as inactive with this update.

Table 9. Active and inactive feature counts for standard AOAs datasets by geography, and service-wide. Count is the same across all AOA types (park, 3km, 30km).

Geography	Active	Inactive	Total
Alaska	16	0	16
American Samoa	1	0	1
Saipan (CNMI)	1	0	1
CONUS	251	0	251
Hawaii	9	0	9
Guam	1	0	1
U.S. Virgin Islands	2	0	2
Combined (service-wide)	281	0	281

Additional Processing

To create the Landscape Dynamics Standard AOAs dataset (geographies) and the associated Combination Landscape Dynamics Standard AOAs dataset (service-wide), deviations from the published SOPs were required. To format data attributes consistently across the output datasets through the application of database domains, the scripts noted in Appendix D were run after all other processing was complete.

Code availability

See Appendices B, C, and D for the Python toolbox scripts used to execute these methods. See Appendix E for session and version information.

Data Records

Two datasets were generated as a part of this effort (Table 10):

1. **Landscape Dynamics Standard AOAs (IMD_BND_ESProtocol_Park3km30km_AOA.gdb).**

This file geodatabase contains feature datasets by geography for each AOA type (park, 3km, 30km), and is available at: <https://irma.nps.gov/DataStore/Reference/Profile/2266452>. Each feature dataset contains one feature class for each park within the geography and one combined feature class of all parks within the geography.

- a. Alaska 30km AOAs (IMD_BND_ESProtocol_Alaska_30km), a feature dataset containing polygon feature classes representing 16 30km-buffered AOAs for Alaska. This feature class is spatially projected to Alaska Albers Equal Area Conic (WKID 3338).
- b. Alaska 3km AOAs (IMD_BND_ESProtocol_Alaska_3km), a feature dataset containing polygon feature classes representing 16 3km-buffered AOAs for Alaska. This feature class is spatially projected to Alaska Albers Equal Area Conic (WKID 3338).
- c. Alaska Park AOAs (IMD_BND_ESProtocol_Alaska_Park), a feature dataset containing polygon feature classes representing 16 park AOAs for Alaska. This feature class is spatially projected to Alaska Albers Equal Area Conic (WKID 3338).
- d. CONUS 30km AOAs (IMD_BND_ESProtocol_CONUS_30km), a feature dataset containing polygon feature classes representing 251 30km-buffered AOAs for CONUS. This feature class is spatially projected to USA Contiguous Albers Equal Area Conic USGS (WKID 102039).
- e. CONUS 3km AOAs (IMD_BND_ESProtocol_CONUS_3km), a feature dataset containing polygon feature classes representing 251 3km-buffered AOAs for CONUS. This feature class is spatially projected to USA Contiguous Albers Equal Area Conic USGS (WKID 102039).
- f. CONUS Park AOAs (IMD_BND_ESProtocol_CONUS_Park), a feature dataset containing polygon feature classes representing 251 park AOAs for CONUS. This feature class is spatially projected to USA Contiguous Albers Equal Area Conic USGS (WKID 102039).
- g. Hawaii 30km AOAs (IMD_BND_ESProtocol_Hawaii_30km), a feature dataset containing polygon feature classes representing nine 30km-buffered AOAs for Hawaii. This feature class is spatially projected to UTM Zone 5N (WKID 26905).
- h. Hawaii 3km AOAs (IMD_BND_ESProtocol_Hawaii_3km), a feature dataset containing polygon feature classes representing nine 3km-buffered AOAs for Hawaii. This feature class is spatially projected to UTM Zone 5N (WKID 26905).
- i. Hawaii Park AOAs (IMD_BND_ESProtocol_Hawaii_Park), a feature dataset containing polygon feature classes representing nine park AOAs for Hawaii. This feature class is spatially projected to UTM Zone 5N (WKID 26905).
- j. Guam 30km AOAs (IMD_BND_ESProtocol_Pacific_Guam_30km), a feature dataset containing polygon feature classes representing one 30km-buffered AOA for Guam. This feature class is spatially projected to UTM Zone 55N (WKID 32655).

- k. Guam 3km AOAs (IMD_BND_ESProtocol_Pacific_Guam_3km), a feature dataset containing polygon feature classes representing one 3km-buffered AOA for Guam. This feature class is spatially projected to UTM Zone 55N (WKID 32655).
- l. Guam Park AOAs (IMD_BND_ESProtocol_Pacific_Guam_Park), a feature dataset containing polygon feature classes representing one park AOA for Guam. This feature class is spatially projected to UTM Zone 55N (WKID 32655).
- m. Saipan 30km AOAs (IMD_BND_ESProtocol_Pacific_Saipan_30km), a feature dataset containing polygon feature classes representing one 30km-buffered AOA for Commonwealth Northern Mariana Islands (Saipan). This feature class is spatially projected to UTM Zone 55N (WKID 32655).
- n. Saipan 3km AOAs (IMD_BND_ESProtocol_Pacific_Saipan_3km), a feature dataset containing polygon feature classes representing one 3km-buffered AOA for Commonwealth Northern Mariana Islands (Saipan). This feature class is spatially projected to UTM Zone 55N (WKID 32655).
- o. Saipan Park AOAs (IMD_BND_ESProtocol_Pacific_Saipan_Park), a feature dataset containing polygon feature classes representing one park AOA for Commonwealth Northern Mariana Islands (Saipan). This feature class is spatially projected to UTM Zone 55N (WKID 32655).
- p. American Samoa 30km AOAs (IMD_BND_ESProtocol_Pacific_Samoa_30km), a feature dataset containing polygon feature classes representing one 30km-buffered AOA for American Samoa. This feature class is spatially projected to UTM Harn Zone 2S (WKID 2195).
- q. American Samoa 3km AOAs (IMD_BND_ESProtocol_Pacific_Samoa_3km), a feature dataset containing polygon feature classes representing one 3km-buffered AOA for American Samoa. This feature class is spatially projected to UTM Harn Zone 2S (WKID 2195).
- r. American Samoa Park AOAs (IMD_BND_ESProtocol_Pacific_Samoa_Park), a feature dataset containing polygon feature classes representing one park AOA for American Samoa. This feature class is spatially projected to UTM Harn Zone 2S (WKID 2195).
- s. U.S. Virgin Islands 30km AOAs (IMD_BND_ESProtocol_USVI_30km), a feature dataset containing polygon feature classes representing two 30km-buffered AOAs for U.S. Virgin Islands. This feature class is spatially projected to USA Contiguous Albers Equal Area Conic USGS (WKID 102039).
- t. U.S. Virgin Islands 3km AOAs (IMD_BND_ESProtocol_USVI_3km), a feature dataset containing polygon feature classes representing two 3km-buffered AOAs for U.S. Virgin Islands. This feature class is spatially projected to USA Contiguous Albers Equal Area Conic USGS (WKID 102039).
- u. U.S. Virgin Islands Park AOAs (IMD_BND_ESProtocol_USVI_Park), a feature dataset containing polygon feature classes representing two park AOAs for U.S. Virgin Islands. This feature class is spatially projected to USA Contiguous Albers Equal Area Conic USGS (WKID 102039).

2. **Combination Landscape Dynamics Standard AOAs**

(IMD_BND_ESProtocol_Park3km30kmCombo_AOA.gdb). This file geodatabase contains two feature classes for each AOA type (park, 3km, 30km), each in a different spatial reference, and available at: <https://irma.nps.gov/DataStore/Reference/Profile/2271955>. Each feature class is a combination of all polygon features for the AOA type service-wide (including Alaska, CONUS, Hawaii, Guam, Saipan, American Samoa, US Virgin Islands, and Puerto Rico).

- a. Landscape Dynamics 30km AOAs (IMD_BND_ESProtocol_30km_albers_py), a polygon feature class representing all 281 30km-buffered AOAs. This feature class is spatially projected to USA Contiguous Albers Equal Area Conic USGS (WKID 102039).
- b. Landscape Dynamics 3km AOAs (IMD_BND_ESProtocol_3km_albers_py), a polygon feature class representing all 281 3km-buffered AOAs. This feature class is spatially projected to USA Contiguous Albers Equal Area Conic USGS (WKID 102039).
- c. Landscape Dynamics Park AOAs (IMD_BND_ESProtocol_Park_albers_py), a polygon feature class representing all 281 park AOAs. This feature class is spatially projected to USA Contiguous Albers Equal Area Conic USGS (WKID 102039).
- d. Landscape Dynamics 30km AOAs (IMD_BND_ESProtocol_30km_wgs_py), a polygon feature class representing all 281 30km-buffered AOAs. This feature class is geographically referenced to the WGS84 datum and not spatially projected (WKID 4326).
- e. Landscape Dynamics 3km AOAs (IMD_BND_ESProtocol_3km_wgs_py), a polygon feature class representing all 281 3km-buffered AOAs. This feature class is geographically referenced to the WGS84 datum and not spatially projected (WKID 4326).
- f. Landscape Dynamics Park AOAs (IMD_BND_ESProtocol_Park_wgs_py), a polygon feature class representing all 281 park AOAs. This feature class is geographically referenced to the WGS84 datum and not spatially projected (WKID 4326).

Table 10. Data sets and their provenance. Temporal currency of datasets is based on the date of publication of Land Resources Division official boundaries. Data sets in italics are not published.

Source Data	Resultant Data	Temporal Currency	Number of Feature Classes	Number of Records ¹
Alternate Boundaries (NPS IMD 2018a) and I&M Operational Boundaries (NPS IMD 2018b)	Interim Landscape Dynamics Boundaries (Not Published)	09-30-2018	1	282
Interim Landscape Dynamics Boundaries	Landscape Dynamics Standard AOAs	09-30-2018	864	1-252
Landscape Dynamics Standard AOAs	Combination Landscape Dynamics Standard AOAs	09-30-2018	6	281

¹The number of records in a feature class will vary

Data Quality Evaluation

Spatial and attribute quality is not quantified in the source data (NPS IMD 2018a) used to produce the landscape dynamics standard AOAs output. The use of alternate boundaries based on local expertise to generate landscape dynamics boundaries and resulting AOAs is considered a quality control process whereby official boundaries are corrected to represent park boundaries as of 9/30/2018. The resultant output datasets are considered fit for use by the NPS Inventory and Monitoring Division when deriving area of analysis-based statistics for the National Park Environmental Settings Monitoring Protocol and inventory studies conducted as a part of the NPS I&M Program.

Usage Notes

The AOA datasets are intended and suitable for use in the collection, analysis, and reporting of metric statistics for the National Park Environmental Setting Monitoring Protocol, or for use in visualization products like web map applications. AOAs may be used in other work by the NPS I&M Program, or Inventory and Monitoring Division, including inventories. These AOAs are not official boundaries and are not intended for use outside NPS Inventory and Monitoring.

References

- DeVivo, J. C., L. Nelson, M. Kineth, T. Philippi, and W. B. Monahan. 2018. Protocol for monitoring environmental setting of National Park Service units: landscape dynamics, climate, and hydrology. Natural Resource Report NPS/IMD/NRR—2018/1844. *National Park Service*, Fort Collins, Colorado. *National Park Service Data Store* <https://irma.nps.gov/DataStore/Reference/Profile/2258457>
- Kineth, M. and L. Nelson. 2018a. NPS unit boundary processing: Boundaries for use for analysis and reporting as a part of the IMD Environmental Settings Protocol. Inventory & Monitoring Division Standard Operating Procedure NPS/IMD/SOP—2.11.5. National Park Service, Fort Collins, Colorado. *National Park Service*, <https://irma.nps.gov/DataStore/Reference/Profile/2256760>
- Kineth, M. and L. Nelson. 2018b. NPS Unit Boundary Processing: Boundary-derived areas of analysis for Use as a Part of the IMD Environmental Settings Protocol Inventory & Monitoring Division Standard Operating Procedure NPS/IMD/SOP—2.11.6. National Park Service, Fort Collins, Colorado. *National Park Service*, <https://irma.nps.gov/DataStore/Reference/Profile/2256760>
- Kineth, M., and L. Nelson. 2020. Unit Boundary Processing for National Park Service Units for use in the National Park Environmental Settings Monitoring Protocol: Fall 2018. Natural Resource Data Series NPS/NRSS/NRDS-2020/1253. National Park Service, Fort Collins, Colorado. *National Park Service Data Store* <https://irma.nps.gov/DataStore/Reference/Profile/2270597>
- NPS Natural Resource Inventory and Monitoring (NPS IMD). 2017. Area of Analysis Polygons - NPS Boundary-Derived (Park, 3km, and 30km). *National Park Service Data Store* <https://irma.nps.gov/DataStore/Reference/Profile/2246123>
- National Park Service Inventory and Monitoring (NPS IMD). 2018a. Unit Boundary Processing – Affiliated Areas and Local Boundary Adjustments. *National Park Service Data Store* <https://irma.nps.gov/DataStore/Reference/Profile/2256634>
- National Park Service Inventory and Monitoring (NPS IMD). 2018b. I&M Operational Boundaries – NPS I&M Inventory and Monitoring (I&M) Boundaries. *National Park Service Data Store* <https://irma.nps.gov/DataStore/Reference/Profile/2266200>
- NPS Natural Resource Inventory and Monitoring (NPS IMD). 2018c. Environmental Settings Protocol – Standard Areas of Analysis (AOAs). *National Park Service Data Store* <https://irma.nps.gov/DataStore/Reference/Profile/2266452>
- NPS Natural Resource Inventory and Monitoring (NPS IMD). 2018d. Environmental Settings Protocol – Combination Standard Areas of Analysis (AOAs). *National Park Service Data Store* <https://irma.nps.gov/DataStore/Reference/Profile/2271955>
- National Park Service Inventory & Monitoring Division (NPS IMD). 2018d. Geodatabase domain tables used in Inventory and Monitoring Division unit boundary processing. *National Park Service Data Store* <https://irma.nps.gov/DataStore/Reference/Profile/2267506>

National Park Service (NPS IRMA). 2018. IRMA Unit Database and visualization feature update.
National Park Service Data Store <https://irma.nps.gov/DataStore/Reference/Profile/2256640>

National Park Service Land Resources Division (NPS LRD). 2018. Administrative Boundaries of
National Park System Units 9/30/2018 - National Geospatial Data Asset (NGDA) NPS National
Parks Dataset. NPS - Land Resources Division. *National Park Service Data Store*
<https://irma.nps.gov/DataStore/Reference/Profile/2225713>

Appendix A. Park AOA Geographies

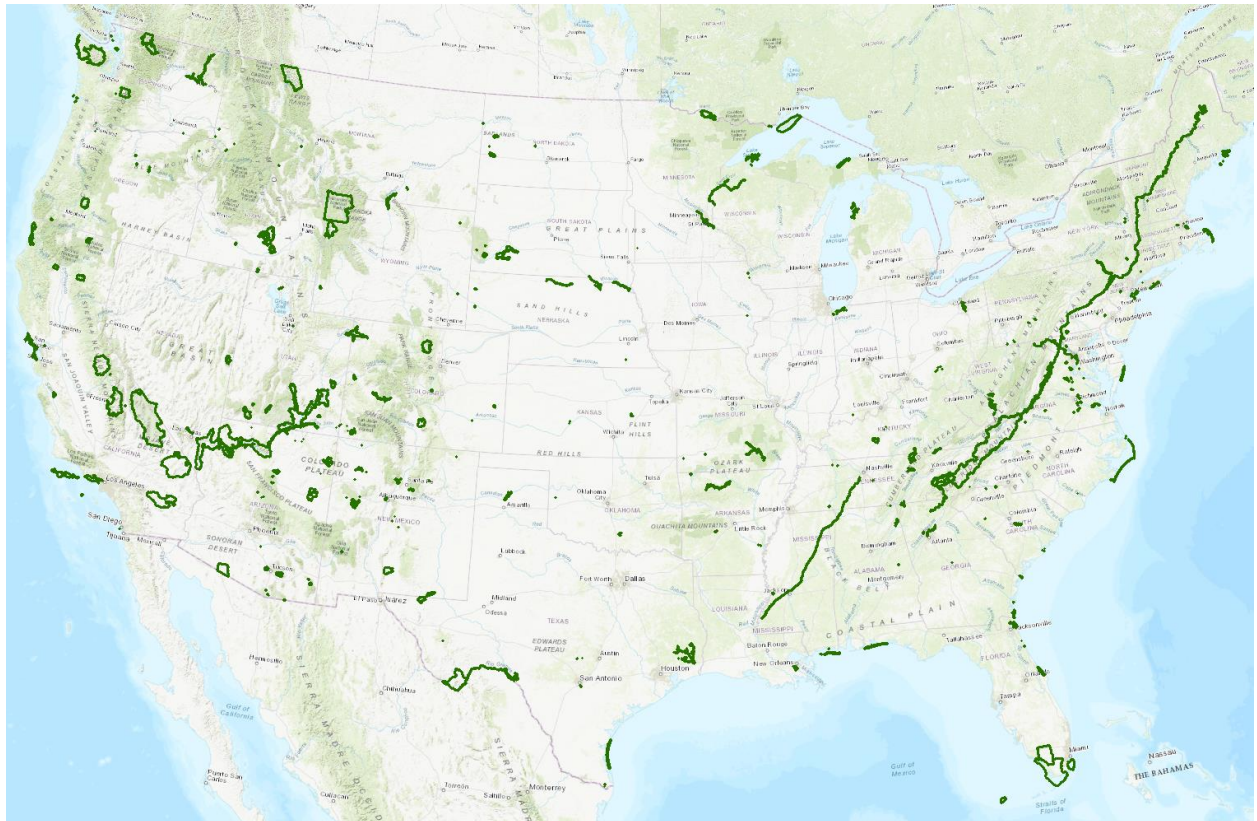


Figure A-1. Spatial extent of park AOAs (in green) in the Continental United States.

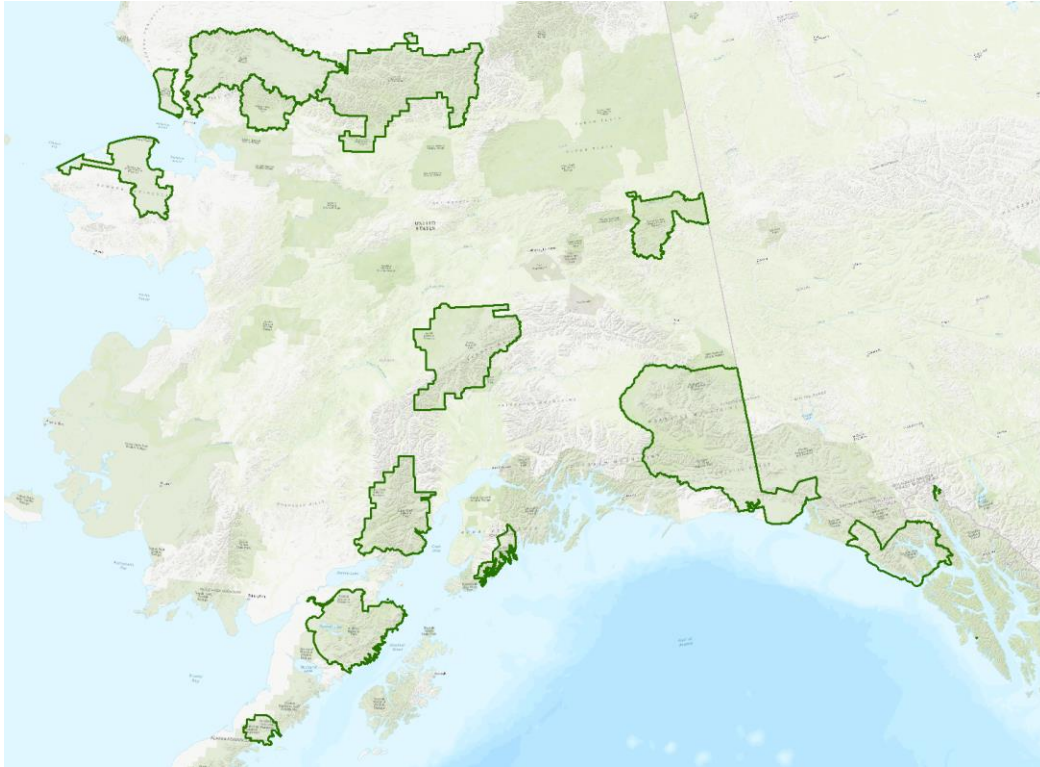


Figure A-2. Spatial extent of park AOAs in Alaska (in green).

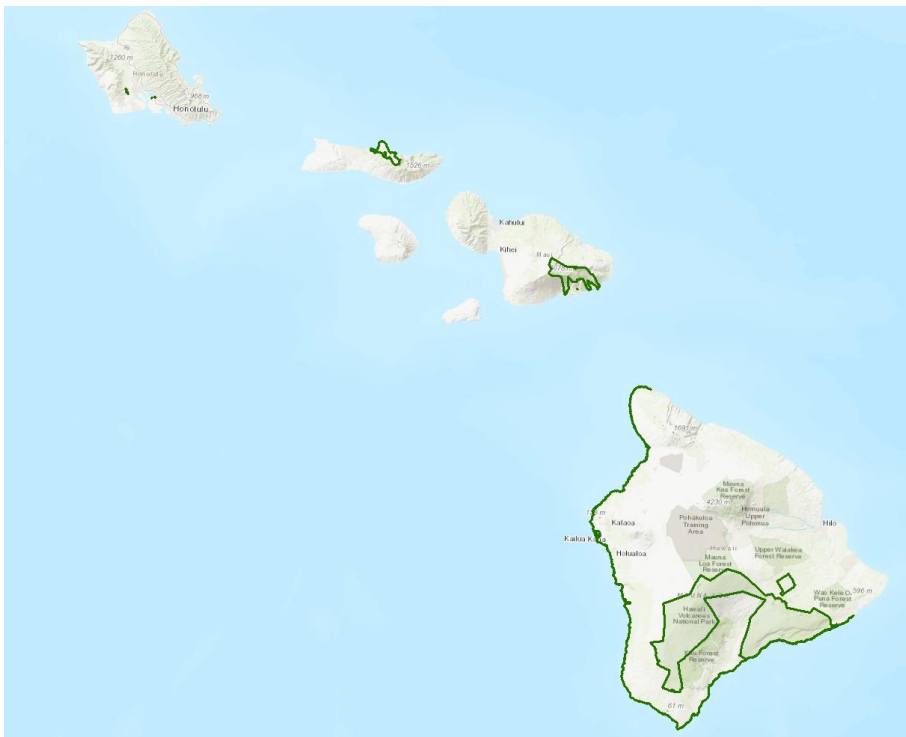


Figure A-3. Spatial extent of park AOAs in Hawaii (in green).

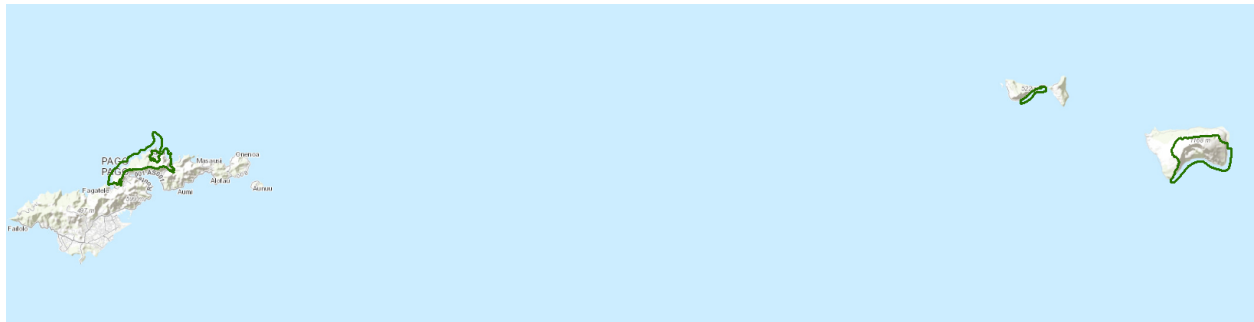


Figure A-4. Spatial extent of park AOAs in American Samoa (in green).

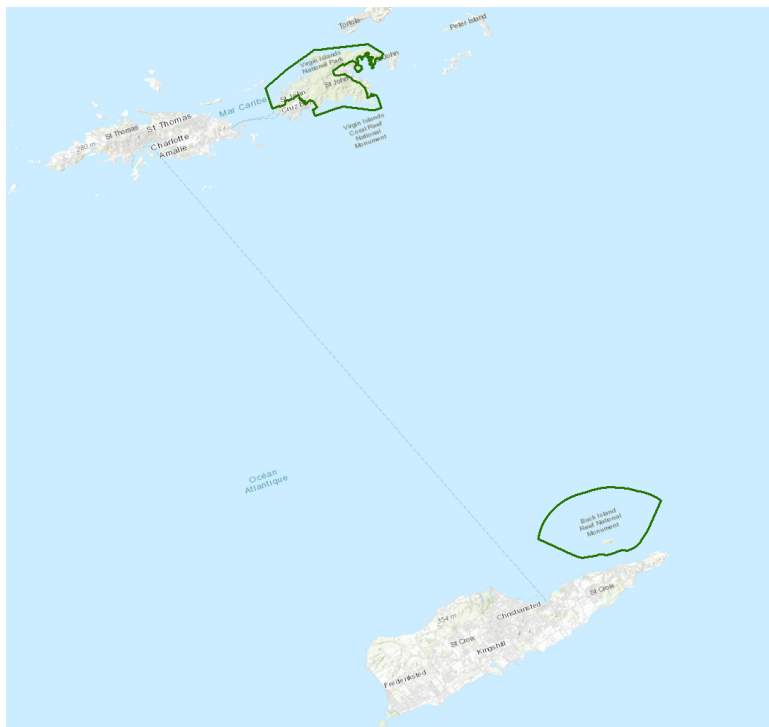


Figure A-5. Spatial extent of park AOAs in U.S. Virgin Islands (in green).

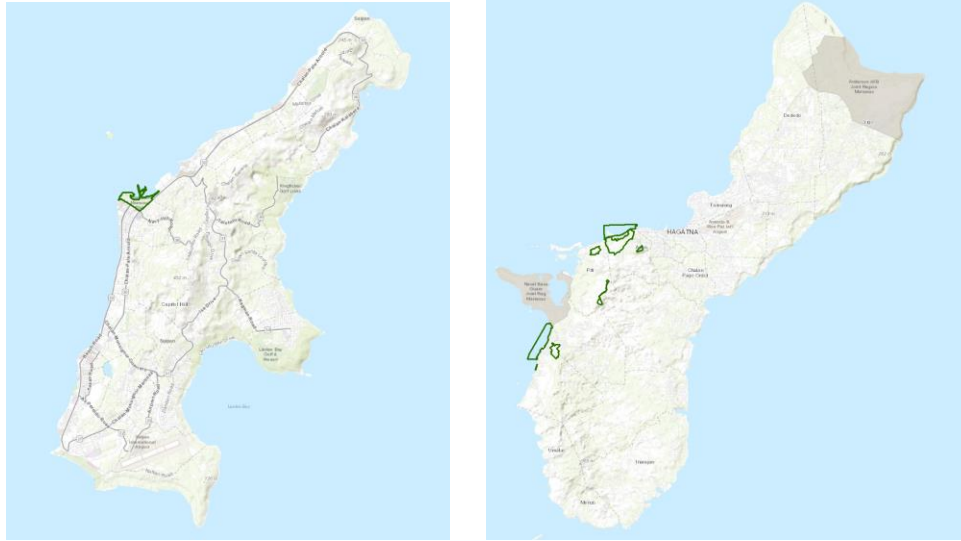


Figure A-6. Spatial extent of park AOAs (in green) in Saipan (left) and Guam (right).

Appendix B. Python Toolbox Code to Create Landscape Dynamics Bounds

The following includes python toolbox code for **Replace Alternate Landscape Dynamics Bounds** and **Update Landscape Dynamics Bounds**.

```
import arcpy
import os, time
from datetime import datetime

class Toolbox(object):
    def __init__(self):
        """Define the toolbox (the name of the toolbox is the name of the
        .pyt file)."""
        self.label = "LandscapeDynamicsBounds_Processing"
        self.alias = ""

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

class AltLDBounds(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "1 Replace Alternate Landscape Dynamics Bounds"
        self.description = "Replace IMD Bounds with Network/Park requested
        Alternate Bounds."
        self.canRunInBackground = False

    def getParameterInfo(self):
        """Define parameter definitions"""

        param0 = arcpy.Parameter(
            displayName = "Feature(s) to Add (GRTE-JODR)",
            name = "GRYN",
            datatype = "GPFeatureLayer",
            parameterType = "Required",
            direction = "Input")

        param1 = arcpy.Parameter(
            displayName = "IMD_BND_Units.gdb",
            name = "fcs",
            datatype = "DEWorkspace",
            parameterType = "Required",
            direction = "Input")

        params = [param0,param1]
        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 execute(self, parameters, messages):
    message = ""
    today=datetime.now()
    datestamp = str(today.isoformat()).replace('-',')[0:8]

    inputAlbers = os.path.join(parameters[1].valueAsText,
"IMD_BND_Units_albers_py")

    GRTE_replace = parameters[0].valueAsText

    Albers =
os.path.join(parameters[1].valueAsText,"imd_unit_bounds_albers_py_TEMP")
    GRTE_temp = os.path.join(parameters[1].valueAsText,"prj_TEMP")

    outSRAlbers =
'PROJCS["USA_Contiguous_Albers_Equal_Area_Conic_USGS_version",GEOGCS["GCS_
North_American_1983",DATUM["D_North_American_1983",SPHEROID["GRS_1980",637
8137.0,298.257222101]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.01745329251
99433]],PROJECTION["Albers"],PARAMETER["False_Easting",0.0],PARAMETER["Fal
se_Northing",0.0],PARAMETER["Central_Meridian",-
96.0],PARAMETER["Standard_Parallel_1",29.5],PARAMETER["Standard_Parallel_2
",45.5],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",6378137.0,29
8.257223563]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]]"
,

    transformMethod = "WGS_1984_(ITRF00)_To_NAD_1983"

    arcpy.AddMessage("Create TEMP feature classes...")
    arcpy.CopyFeatures_management(inputAlbers, Albers);
messages.addGPMessages()

```

```

arcpy.CalculateField_management(Albers,"UnitCodeQualifier","SC","PYTHON"
); messages.addGPMessages()

arcpy.CalculateField_management(Albers,"FeatureQualifier","NC","PYTHON")
; messages.addGPMessages()

        arcpy.AddMessage("Add GRTE-JODR merged feature to Albers...")
        arcpy.MakeFeatureLayer_management(Albers,"Albers");
messages.addGPMessages()

arcpy.SelectLayerByAttribute_management("Albers","NEW_SELECTION","UNIT_COD
E = 'GRTE'"); messages.addGPMessages()

arcpy.CalculateField_management("Albers","Lifecycle","Inactive","PYTHON"
); messages.addGPMessages()

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

arcpy.Project_management(GRTE_replace,GRTE_temp,outSRAlbers,transformMetho
d); messages.addGPMessages()
        arcpy.CalculateField_management(GRTE_temp,"Identifier","IMD-Park-
102039-!Unit_Code_Display!-!CreateDate!","PYTHON");
messages.addGPMessages()
        arcpy.Append_management([GRTE_temp],"Albers","NO_TEST");
messages.addGPMessages()
        arcpy.Delete_management(GRTE_temp); messages.addGPMessages()

class UpdateLandscapeDynamicsBounds(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "2 Update Landscape Dynamics Bounds"
        self.description = "Create or update Landscape Dynamics Bounds
feature classes in LandscapeDynamics_Bounds.gdb"
        self.canRunInBackground = False

    def getParameterInfo(self):
        """Define parameter definitions"""

        param0 = arcpy.Parameter(
            displayName = "Source IMD Bounds Feature Class - Albers",
            name = "albersBounds",
            datatype = "GPFeatureLayer",
            parameterType = "Required",
            direction = "Input")

```

```

        param0.value =
r"X:\ProjectData\Data_Processing\Bounds_Processing\Data\IMD_Bounds.gdb\imd
_unit_bounds_albers_py_TEMP"

    param1 = arcpy.Parameter(
        displayName = "Output GeoDatabase",
        name = "outDB",
        datatype = "DEWorkspace",
        parameterType = "Required",
        direction = "Input")

    param2 = arcpy.Parameter(
        displayName = "Landscape Dynamics Parks Lookup Table",
        name = "altBounds",
        datatype = "DETable",
        parameterType = "Required",
        direction = "Input")
    param2.value = ""

    param3 = arcpy.Parameter(
        displayName = "Folder Containing Metadata Templates",
        name = "metaFolder",
        datatype = "Folder",
        parameterType = "Required",
        direction = "Input")
    param3.value = ""

    params = [param0, param1, param2, param3]
    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 projectDataset(self, sourceDataset, outpuDataset,
outputSpatialRef, transform=None):

```

```

        """
        Re-projects and repairs source dataset
        """
        arcpy.Project_management(sourceDataset, outputDataset,
outputSpatialRef, transform)
        arcpy.RepairGeometry_management(outputDataset)
        arcpy.AddMessage('\n Re-projected to ' + outputDataset)

    def populateDatasets(self, targetLayer, targetDataset, selectClause,
outputFC):
        arcpy.SelectLayerByAttribute_management(targetLayer,
"CLEAR_SELECTION")
        arcpy.SelectLayerByAttribute_management(targetLayer,
"NEW_SELECTION", "UNIT_CODE IN (" + selectClause + ")")
        arcpy.CopyFeatures_management(targetLayer, outputFC)
        if targetDataset == "Alaska":
arcpy.CalculateField_management(outputFC, "Identifier", "IMD-LDPark-3338-
"!Unit_Code_Display!" - "!CreateDate!", "PYTHON")
            elif targetDataset == "Hawaii":
arcpy.CalculateField_management(outputFC, "Identifier", "IMD-LDPark-26905-
"!Unit_Code_Display!" - "!CreateDate!", "PYTHON")
            elif targetDataset == "Pacific_Saipan_Guam":
arcpy.CalculateField_management(outputFC, "Identifier", "IMD-LDPark-32655-
"!Unit_Code_Display!" - "!CreateDate!", "PYTHON")
            elif targetDataset == "Pacific_Samoa":
arcpy.CalculateField_management(outputFC, "Identifier", "IMD-LDPark-2195-
"!Unit_Code_Display!" - "!CreateDate!", "PYTHON")
            else: pass
        arcpy.SelectLayerByAttribute_management(targetLayer,
"CLEAR_SELECTION")
        arcpy.RepairGeometry_management(outputFC)
        arcpy.AddMessage('\n Created ' + outputFC)

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

        arcpy.env.overwriteOutput = 1
        # Lands bounds have M and Z values for some reason; this is an
extra check
        # since M and Z are disabled in the UnitBounds_Processing
UpdateUnitBounds tool
        arcpy.env.outputMFlag = "Disabled"
        arcpy.env.outputZFlag = "Disabled"
        outputInitTempFeatureClassName = "temp_npscape_bounds0"
        outputTempFeatureClassName = "temp_npscape_bounds"
        outputGDBFeatureClassName = "IMD_LD_BND"

```

```

tempLayer = "tempLayer"
lookupTable = "lookupTable"
tableName = parameters[2].valueAsText
selectClause = "UNIT_CODE IN (SELECT Code FROM " +lookupTable +
") "

Albers = parameters[0].valueAsText
WGSWeb = parameters[1].valueAsText
outputGDBFeatureClassInitTemp =
os.path.join(parameters[1].valueAsText, outputInitTempFeatureClassName)
outputGDBFeatureClassTemp =
os.path.join(parameters[1].valueAsText, outputTempFeatureClassName)
outputGDBFeatureClassAlbers =
os.path.join(parameters[1].valueAsText, outputGDBFeatureClassName +
"_albers")
outputGDBFeatureClassWGS = os.path.join(parameters[1].valueAsText,
outputGDBFeatureClassName + "_wgs")
WebMerc = os.path.join(parameters[1].valueAsText,
outputGDBFeatureClassName + "_webmercator_TEMP")
outputGDBFeatureClassWebMerc =
os.path.join(parameters[1].valueAsText, outputGDBFeatureClassName +
"_webmercator")
outputGDBFeatureClassConvexHulls =
os.path.join(parameters[1].valueAsText, outputGDBFeatureClassName +
"_webmercator_convexhulls")

outputGDBFeatureClassAlaska =
os.path.join(parameters[1].valueAsText, "Alaska\\" +
outputGDBFeatureClassName + "_Alaska")
outputGDBFeatureClassHawaii =
os.path.join(parameters[1].valueAsText, "Hawaii\\" +
outputGDBFeatureClassName + "_Hawaii")
outputGDBFeatureClassSaipan =
os.path.join(parameters[1].valueAsText, "Pacific_Saipan_Guam\\" +
outputGDBFeatureClassName + "_Saipan_Guam")
outputGDBFeatureClassSamoa =
os.path.join(parameters[1].valueAsText, "Pacific_Samoa\\" +
outputGDBFeatureClassName + "_Samoa")
outputGDBFeatureClassCONUS =
os.path.join(parameters[1].valueAsText, "CONUS\\" +
outputGDBFeatureClassName + "_CONUS")

# Note: AllUnits array is never used
AllUnits = []
cursor = arcpy.SearchCursor(parameters[2].valueAsText)
for row in cursor:
    AllUnits.append(row.getValue("Code"))

```

```

        Alaska_Parks =
        "'ALAG','ANIA','BELA','CAKR','DENA','GAAR','GLBA','KATM','KEFJ','KOVA','KL
        GO','LACL','NOAT','SITK','YUCH','WRST'"
        Hawaii_Parks =
        "'ALKA','HALE','KALA','KAHO','HAVO','HONO','PUHE','PUHO','VALR'"
        PacificSaipan_Parks = "'AMME','WAPA'"
        PacificSamoa_Parks= "'NPSA'"
        Alaska_Parks_Array =
        ['ALAG','ANIA','BELA','CAKR','DENA','GAAR','GLBA','KATM','KEFJ','KOVA','KL
        GO','LACL','NOAT','SITK','YUCH','WRST']
        Hawaii_Parks_Array =
        ['ALKA','HALE','KALA','KAHO','HAVO','HONO','PUHE','PUHO','VALR']
        PacificSaipan_Parks_Array = ['AMME','WAPA']
        PacificSamoa_Parks_Array = ['NPSA']

        # Delete older versions
        fcsToDelete = [outputGDBFeatureClassName + '_albers_2*',
                        outputGDBFeatureClassName + '_webmercator_2*',
                        outputGDBFeatureClassName +
        '_webmercator_convexhulls_2*']

        workspace = parameters[1].valueAsText
        arcpy.env.workspace = workspace
        for fc in fcsToDelete:
            if arcpy.Exists(fc):
                fcs = arcpy.ListFeatureClasses(fc, "Polygon")
                for fcc in fcs:
                    arcpy.Delete_management(fcc); messages.addGPMessages()

        # Version existing feature classes; Project tp new feature classes
        fcsToVersion = [outputGDBFeatureClassAlbers,
                        outputGDBFeatureClassWGS,
                        outputGDBFeatureClassWebMerc,
                        outputGDBFeatureClassConvexHulls]

        for fc in fcsToVersion:
            if arcpy.Exists(fc):
                arcpy.Rename_management(fc, fc + '_' + datestamp);
messages.addGPMessages()

        outSRWebMerc =
        'PROJCS["WGS_1984_Web_Mercator_Auxiliary_Sphere",GEOGCS["GCS_WGS_1984",DAT
        UM["D_WGS_1984",SPHEROID["WGS_1984",6378137.0,298.257223563]],PRIMEM["Gree
        nwich",0.0],UNIT["Degree",0.017453292519943295]],PROJECTION["Mercator_Auxi
        liary_Sphere"],PARAMETER["False_Easting",0.0],PARAMETER["False_Northing",0
        .0],PARAMETER["Central_Meridian",0.0],PARAMETER["Standard_Parallel_1",0.0]
        ,PARAMETER["Auxiliary_Sphere_Type",0.0],UNIT["Meter",1.0]]'

```

```

        # Join lookup to unit_bounds_wgs and copy matching features to
temp feature class
        arcpy.FeatureClassToFeatureClass_conversion(Albers,
parameters[1].valueAsText, outputInitTempFeatureClassName);
messages.addGPMessages()

arcpy.CalculateField_management(outputGDBFeatureClassInitTemp,"DSLink","h
tps://irma.nps.gov/DataStore/Reference/Profile/2235933?lnv=True")
        arcpy.MakeFeatureLayer_management(outputGDBFeatureClassInitTemp,
tempLayer); messages.addGPMessages()
        arcpy.CopyRows_management(tableName,lookupTable);
messages.addGPMessages()
        arcpy.SelectLayerByAttribute_management(tempLayer,"NEW_SELECTION",
selectClause); messages.addGPMessages()

        arcpy.CopyFeatures_management(tempLayer,
outputGDBFeatureClassAlbers); messages.addGPMessages()

arcpy.SelectLayerByAttribute_management(tempLayer,"CLEAR_SELECTION");
messages.addGPMessages()
        arcpy.RepairGeometry_management(outputGDBFeatureClassAlbers);
messages.addGPMessages()

        arcpy.CopyFeatures_management(tempLayer,
outputGDBFeatureClassTemp); messages.addGPMessages()
        arcpy.RepairGeometry_management(outputGDBFeatureClassTemp);
messages.addGPMessages()

        # Create feature classes in feature datasets:
        arcpy.Delete_management(tempLayer); messages.addGPMessages()
        arcpy.MakeFeatureLayer_management(outputGDBFeatureClassTemp,
tempLayer); messages.addGPMessages()

        UpdateLandscapeDynamicsBounds.populateDatasets(self, tempLayer,
"Alaska", Alaska_Parks, outputGDBFeatureClassAlaska)
        UpdateLandscapeDynamicsBounds.populateDatasets(self, tempLayer,
"Hawaii", Hawaii_Parks, outputGDBFeatureClassHawaii)
        UpdateLandscapeDynamicsBounds.populateDatasets(self, tempLayer,
"Pacific_Saipan_Guam", PacificSaipan_Parks, outputGDBFeatureClassSaipan)
        UpdateLandscapeDynamicsBounds.populateDatasets(self, tempLayer,
"Pacific_Samoa", PacificSamoa_Parks, outputGDBFeatureClassSamoa)

        arcpy.SelectLayerByAttribute_management(tempLayer,
"NEW_SELECTION", "UNIT_CODE IN (" + Alaska_Parks + ")");
messages.addGPMessages()
        arcpy.SelectLayerByAttribute_management(tempLayer,
"ADD_TO_SELECTION", "UNIT_CODE IN (" + Hawaii_Parks + ")");
messages.addGPMessages()

```

```

        arcpy.SelectLayerByAttribute_management(tempLayer,
"ADD_TO_SELECTION", "UNIT_CODE IN (" + PacificSaipan_Parks + ")");
messages.addGPMessages()
        arcpy.SelectLayerByAttribute_management(tempLayer,
"ADD_TO_SELECTION", "UNIT_CODE IN (" + PacificSamoa_Parks + ")");
messages.addGPMessages()
        arcpy.SelectLayerByAttribute_management(tempLayer,
"SWITCH_SELECTION"); messages.addGPMessages()
        arcpy.CopyFeatures_management(tempLayer,
outputGDBFeatureClassCONUS)
        arcpy.SelectLayerByAttribute_management(tempLayer,
"CLEAR_SELECTION")
        arcpy.RepairGeometry_management(outputGDBFeatureClassCONUS)

itemsToDelete = [tempLayer, lookupTable,
outputGDBFeatureClassTemp, outputGDBFeatureClassInitTemp, Albers]
for item in itemsToDelete:
    if arcpy.Exists(item):
        arcpy.Delete_management(item); messages.addGPMessages()

```

Appendix C. Python Toolbox Code to Create AOAs

The following includes python toolbox code for **Create Map Service Landscape Dynamics AOAs**

```
import arcpy
import os
import sys
import 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_NewLandscapeDynamicsAOAs"
        self.alias = ""

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

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

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

param3 = arcpy.Parameter(
    displayName = "Landscape Dynamics Park Bounds Lookup Table",
    name = "altBounds",
    datatype = "DETable",
    parameterType = "Required",
    direction = "Input")
param3.value = r""

params = [param0,param1,param2,param3]
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, StatusLookups, MetadataTemplate, ID=None):
    """
    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.JoinField_management(OutputGDB+"\\\\"+FeatureClass,"UNIT_CODE",Lookup,

```

```

"PARK_CODE",["UnitName","NetworkCode","NetworkName","RegionName","PARK_CODE"])

arcpy.JoinField_management(OutputGDB+"\\ "+FeatureClass,"UNIT_CODE",StatusLookups,"UNIT_CODE",["DATE_EDIT","Feature_Source","Source_Metadata","DSLink","Identifier","CreateDate","Full_Name","RedactReason","ChangeReason","UnitCodeQualifier","ProcessingQualifier","DataCertificationLevel"])

    if ID:

arcpy.CalculateField_management(OutputGDB+"\\ "+FeatureClass,"Identifier",ID,"PYTHON")
    arcpy.MetadataImporter_conversion(MetadataTemplate,OutputGDB+"\\ "+FeatureClass)
    arcpy.AddMessage ('\n Created ' + OutputGDB + 'source feature class \n')

    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.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", "")
        if fd == "IMD_BND_ESProtocol_Alaska_30km":
            arcpy.CalculateField_management(OutputGDB + "\\ "+fd+"\\ "+buffFeatureClass,"Identifier","IMD-LDAOA30km-3338-!Unit_Code_Display!-!CreateDate!',"PYTHON")
        elif fd == "IMD_BND_ESProtocol_CONUS_30km":
            arcpy.CalculateField_management(OutputGDB + "\\ "+fd+"\\ "+buffFeatureClass,"Identifier","IMD-LDAOA30km-102039-!Unit_Code_Display!-!CreateDate!',"PYTHON")
        elif fd == "IMD_BND_ESProtocol_Hawaii_30km":
            arcpy.CalculateField_management(OutputGDB + "\\ "+fd+"\\ "+buffFeatureClass,"Identifier","IMD-LDAOA30km-26905-!Unit_Code_Display!-!CreateDate!',"PYTHON")
        elif fd == "IMD_BND_ESProtocol_Pacific_Saipan_30km":
            arcpy.CalculateField_management(OutputGDB + "\\ "+fd+"\\ "+buffFeatureClass,"Identifier","IMD-LDAOA30km-32655-!Unit_Code_Display!-!CreateDate!',"PYTHON")
        elif fd == "IMD_BND_ESProtocol_Pacific_Guam_30km":
            arcpy.CalculateField_management(OutputGDB + "\\ "+fd+"\\ "+

```

```

buffFeatureClass,"Identifier","IMD-LDAOA30km-32655-!Unit_Code_Display!"-
"!CreateDate!',"PYTHON")
    elif fd == "IMD_BND_ESProtocol_Pacific_Samoa_30km":
arcpy.CalculateField_management(OutputGDB + "\\ "+fd+"\\ " +
buffFeatureClass,"Identifier","IMD-LDAOA30km-2195-!Unit_Code_Display!"-
"!CreateDate!',"PYTHON")
    elif fd == "IMD_BND_ESProtocol_USVI_30km":
arcpy.CalculateField_management(OutputGDB + "\\ "+fd+"\\ " +
buffFeatureClass,"Identifier","IMD-LDAOA30km-102039-
"!Unit_Code_Display!"-!CreateDate!',"PYTHON")
    elif fd == "IMD_BND_ESProtocol_Alaska_3km":
arcpy.CalculateField_management(OutputGDB + "\\ "+fd+"\\ " +
buffFeatureClass,"Identifier","IMD-LDAOA3km-3338-!Unit_Code_Display!"-
"!CreateDate!',"PYTHON")
    elif fd == "IMD_BND_ESProtocol_CONUS_3km":
arcpy.CalculateField_management(OutputGDB + "\\ "+fd+"\\ " +
buffFeatureClass,"Identifier","IMD-LDAOA3km-102039-!Unit_Code_Display!"-
"!CreateDate!',"PYTHON")
    elif fd == "IMD_BND_ESProtocol_Hawaii_3km":
arcpy.CalculateField_management(OutputGDB + "\\ "+fd+"\\ " +
buffFeatureClass,"Identifier","IMD-LDAOA3km-26905-!Unit_Code_Display!"-
"!CreateDate!',"PYTHON")
    elif fd == "IMD_BND_ESProtocol_Pacific_Saipan_3km":
arcpy.CalculateField_management(OutputGDB + "\\ "+fd+"\\ " +
buffFeatureClass,"Identifier","IMD-LDAOA3km-32655-!Unit_Code_Display!"-
"!CreateDate!',"PYTHON")
    elif fd == "IMD_BND_ESProtocol_Pacific_Guam_3km":
arcpy.CalculateField_management(OutputGDB + "\\ "+fd+"\\ " +
buffFeatureClass,"Identifier","IMD-LDAOA3km-32655-!Unit_Code_Display!"-
"!CreateDate!',"PYTHON")
    elif fd == "IMD_BND_ESProtocol_Pacific_Samoa_3km":
arcpy.CalculateField_management(OutputGDB + "\\ "+fd+"\\ " +
buffFeatureClass,"Identifier","IMD-LDAOA3km-2195-!Unit_Code_Display!"-
"!CreateDate!',"PYTHON")
    elif fd == "IMD_BND_ESProtocol_USVI_3km":
arcpy.CalculateField_management(OutputGDB + "\\ "+fd+"\\ " +
buffFeatureClass,"Identifier","IMD-LDAOA3km-102039-!Unit_Code_Display!"-
"!CreateDate!',"PYTHON")
    else: pass
arcpy.AddMessage ('\\n Populated ' + buffFeatureClass)

def createBuffer(self,
OutputGDB,FeatureClass,FeatureDataset,Buffer3,Buffer30,fc3km,fc30km,FCName
): #,Suffix3,Suffix30
    """
        Buffer each geography feature class 3km, and 30km
    """
    arcpy.Buffer_analysis(OutputGDB + "\\ " +
FeatureClass,fc3km,Buffer3,"FULL","ROUND","NONE")

```

```

arcpy.Buffer_analysis(OutputGDB + "\\\" +
FeatureClass,fc30km,Buffer30,"FULL","ROUND","NONE")

arcpy.AddField_management(fc30km, "AREA_SQKM", "FLOAT", "11", "3",
"", "", "NULLABLE", "NON_REQUIRED", "")
arcpy.CalculateField_management(fc30km, "AREA_SQKM", "!shape.area!
/ 1000000", "PYTHON", "")
arcpy.JoinField_management(fc30km,"UNIT_CODE",OutputGDB + "\\\" +
'IMD_BND_ESProtocol_Park_albers_py',"UNIT_CODE",["UNIT_NAME"])
if FCName == "Alaska_AOA_Park":
arcpy.CalculateField_management(fc30km,"Identifier","IMD-LDAOA30km-3338-
"!Unit_Code_Display!"-"!CreateDate!',"PYTHON")
elif FCName == "CONUS_AOA_Park":
arcpy.CalculateField_management(fc30km,"Identifier","IMD-LDAOA30km-
102039-"!Unit_Code_Display!"-"!CreateDate!',"PYTHON")
elif FCName == "Hawaii_AOA_Park":
arcpy.CalculateField_management(fc30km,"Identifier","IMD-LDAOA30km-26905-
"!Unit_Code_Display!"-"!CreateDate!',"PYTHON")
elif FCName == "Saipan_AOA_Park":
arcpy.CalculateField_management(fc30km,"Identifier","IMD-LDAOA30km-32655-
"!Unit_Code_Display!"-"!CreateDate!',"PYTHON")
elif FCName == "Guam_AOA_Park":
arcpy.CalculateField_management(fc30km,"Identifier","IMD-LDAOA30km-32655-
"!Unit_Code_Display!"-"!CreateDate!',"PYTHON")
elif FCName == "Samoa_AOA_Park":
arcpy.CalculateField_management(fc30km,"Identifier","IMD-LDAOA30km-2195-
"!Unit_Code_Display!"-"!CreateDate!',"PYTHON")
elif FCName == "USVI_AOA_Park":
arcpy.CalculateField_management(fc30km,"Identifier","IMD-LDAOA30km-
102039-"!Unit_Code_Display!"-"!CreateDate!',"PYTHON")
else: arcpy.CalculateField_management(fc30km,"Identifier","IMD-
LDAOA30km-102039-"!Unit_Code_Display!"-"!CreateDate!',"PYTHON")

arcpy.CalculateField_management(fc30km,"UnitCodeQualifier","SC","PYTHON"
)

arcpy.CalculateField_management(fc30km,"FeatureQualifier","NC","PYTHON")

arcpy.AddField_management(fc30km, "AOA", "TEXT", "", "", "10", "",
"NULLABLE", "NON_REQUIRED", "")
arcpy.CalculateField_management(fc30km, "AOA",
"!PARK_CODE!+_30km'", "PYTHON", "")
arcpy.AddField_management(fc3km, "AOA", "TEXT", "", "", "10", "",
"NULLABLE", "NON_REQUIRED", "")
arcpy.CalculateField_management(fc3km, "AOA",
"!PARK_CODE!+_3km'", "PYTHON", "")
if FCName == "Alaska_AOA_Park":
arcpy.CalculateField_management(fc3km,"Identifier","IMD-LDAOA3km-3338-
"!Unit_Code_Display!"-"!CreateDate!',"PYTHON")

```

```

        elif FCName == "CONUS_AOA_Park":
arcpy.CalculateField_management(fc3km,"Identifier","IMD-LDAOA3km-102039-
"!Unit_Code_Display!"-"!CreateDate!',"PYTHON")
        elif FCName == "Hawaii_AOA_Park":
arcpy.CalculateField_management(fc3km,"Identifier","IMD-LDAOA3km-26905-
"!Unit_Code_Display!"-"!CreateDate!',"PYTHON")
        elif FCName == "Saipan_AOA_Park":
arcpy.CalculateField_management(fc3km,"Identifier","IMD-LDAOA3km-32655-
"!Unit_Code_Display!"-"!CreateDate!',"PYTHON")
        elif FCName == "Guam_AOA_Park":
arcpy.CalculateField_management(fc3km,"Identifier","IMD-LDAOA3km-32655-
"!Unit_Code_Display!"-"!CreateDate!',"PYTHON")
        elif FCName == "Samoa_AOA_Park":
arcpy.CalculateField_management(fc3km,"Identifier","IMD-LDAOA3km-2195-
"!Unit_Code_Display!"-"!CreateDate!',"PYTHON")
        elif FCName == "USVI_AOA_Park":
arcpy.CalculateField_management(fc3km,"Identifier","IMD-LDAOA3km-102039-
"!Unit_Code_Display!"-"!CreateDate!',"PYTHON")
        else: arcpy.CalculateField_management(fc3km,"Identifier","IMD-
LDAOA3km-102039-"!Unit_Code_Display!"-"!CreateDate!',"PYTHON")

arcpy.CalculateField_management(fc30km,"UnitCodeQualifier","SC","PYTHON"
)

arcpy.CalculateField_management(fc30km,"FeatureQualifier","NC","PYTHON")

arcpy.CopyFeatures_management(FeatureClass,OutputGDB+"\\\\"+FeatureDataset+"
\\\\" + FCName)
    arcpy.AddMessage('\n Buffered and copied '+ FeatureDataset)

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

    arcpy.env.overwriteOutput = 1
    Output_FeatureClass_Root = "landscape_dynamics_bounds_albers"
    Source_FeatureClass = parameters[0].valueAsText
    Dissolved_FeatureClass_Root =
"LandscapeDynamics_Bounds_Albers_Dissolved"
    Dissolved_FC_3km = "LandscapeDynamics_3km_Dissolved"
    Dissolved_FC_30km = "LandscapeDynamics_30km_Dissolved"

    # Use a copy of the enterprise 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,"IMD_BND_ESProtocol_Park3km30km_AOA
.gdb")
        Versioned_Output_TargetGDB =
os.path.join(parameters[1].valueAsText,
"IMD_BND_ESProtocol_Park3km30km_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 = Output_TargetGDB + "\\ " +
Output_FeatureClass_Root
        Metadata_Template =
r'X:\ProjectData\NPSScape\Templates\Metadata\NPSScape_AOA_Template.xml '

        WGS_SpatialRef = arcpy.SpatialReference(3857)
        #'PROJCS["WGS_1984_Web_Mercator_Auxiliary_Sphere",GEOGCS["GCS_WGS_1984",DA
TUM["D_WGS_1984",SPHEROID["WGS_1984",6378137.0,298.257223563]],PRIMEM["Gre
enwich",0.0],UNIT["Degree",0.017453292519943295]],PROJECTION["Mercator_Aux
iliary_Sphere"],PARAMETER["False_Easting",0.0],PARAMETER["False_Northing",
0.0],PARAMETER["Central_Meridian",0.0],PARAMETER["Standard_Parallel_1",0.0
],PARAMETER["Auxiliary_Sphere_Type",0.0],UNIT["Meter",1.0]] '
        Conus_SpatialRef = arcpy.SpatialReference(102039)
        #'PROJCS["USA_Contiguous_Albers_Equal_Area_Conic_USGS_version",GEOGCS["GCS
_North_American_1983",DATUM["D_North_American_1983",SPHEROID["GRS_1980",63
78137.0,298.257222101]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925
199433]],PROJECTION["Albers"],PARAMETER["False_Easting",0.0],PARAMETER["Fa
lse_Northing",0.0],PARAMETER["Central_Meridian",-
96.0],PARAMETER["Standard_Parallel_1",29.5],PARAMETER["Standard_Parallel_2
",45.5],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",6378137.0,29
8.257223563]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]]")
' #ConusSRDesc.SpatialReference
        AK_SpatialRef = arcpy.SpatialReference(3338)
        #'PROJCS["Alaska_Albers_Equal_Area_Conic",GEOGCS["GCS_North_American_1983"
,DATUM["D_North_American_1983",SPHEROID["GRS_1980",6378137,298.257222101]]
,PRIMEM["Greenwich",0],UNIT["Degree",0.017453292519943295]],PROJECTION["Al
bers"],PARAMETER["False_Easting",0],PARAMETER["False_Northing",0],PARAMETE
R["central_meridian",-
154],PARAMETER["Standard_Parallel_1",55],PARAMETER["Standard_Parallel_2",6
5],PARAMETER["latitude_of_origin",50],UNIT["Meter",1]] '
        #AKSRDesc.SpatialReference

```

```

HawaiiSpatialRef = arcpy.SpatialReference(26905)
#'PROJCS["NAD_1983_UTM_Zone_5N",GEOGCS["GCS_North_American_1983",DATUM["D_North_American_1983",SPHEROID["GRS_1980",6378137,298.257222101]],PRIMEM["Greenwich",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],PARAMETER["false_northing",0],UNIT["Meter",1]]'
#HawaiiSRDesc.SpatialReference

PacificSaipanSpatialRef = arcpy.SpatialReference(32655)
#'PROJCS["WGS_1984_UTM_Zone_55N",GEOGCS["GCS_WGS_1984",DATUM["D_WGS_1984",SPHEROID["WGS_1984",6378137,298.257223563]],PRIMEM["Greenwich",0],UNIT["Degree",0.017453292519943295]],PROJECTION["Transverse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",147],PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_northing",0],UNIT["Meter",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 = arcpy.SpatialReference(2195)
#'PROJCS["NAD83(HARN) / UTM zone 2S",GEOGCS["NAD83(HARN)",DATUM["D_North_American_1983_HARN",SPHEROID["GRS_1980",6378137,298.257222101]],PRIMEM["Greenwich",0],UNIT["Degree",0.017453292519943295]],PROJECTION["Transverse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["central_meridian",-171],PARAMETER["scale_factor",0.9996],PARAMETER["false_easting",500000],PARAMETER["false_northing",10000000],UNIT["Meter",1]]'
#PacificSamoaSRDesc.SpatialReference

USVISpatialRef = arcpy.SpatialReference(102039)
#'PROJCS["USA_Contiguous_Albers_Equal_Area_Conic_USGS_version",GEOGCS["GCS_North_American_1983",DATUM["D_North_American_1983",SPHEROID["GRS_1980",6378137.0,298.257222101]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]],PROJECTION["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",45.5],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",6378137.0,298.257223563]],PRIMEM["Greenwich",0.0],UNIT["Degree",0.0174532925199433]]')
,

Conus_FeatureDataset_Park = "IMD_BND_ESProtocol_CONUS_Park"
AK_FeatureDataset_Park = "IMD_BND_ESProtocol_Alaska_Park"
Hawaii_FeatureDataset_Park = "IMD_BND_ESProtocol_Hawaii_Park"
PacificSaipan_FeatureDataset_Park =
"IMD_BND_ESProtocol_Pacific_Saipan_Park"
PacificGuam_FeatureDataset_Park =
"IMD_BND_ESProtocol_Pacific_Guam_Park"
PacificSamoa_FeatureDataset_Park =
"IMD_BND_ESProtocol_Pacific_Samoa_Park"

```

```

USVI_FeatureDataset_Park = "IMD_BND_ESProtocol_USVI_Park"
Conus_FeatureDataset_3km = "IMD_BND_ESProtocol_CONUS_3km"
AK_FeatureDataset_3km = "IMD_BND_ESProtocol_Alaska_3km"
Hawaii_FeatureDataset_3km = "IMD_BND_ESProtocol_Hawaii_3km"
PacificSaipan_FeatureDataset_3km =
"IMD_BND_ESProtocol_Pacific_Saipan_3km"
PacificGuam_FeatureDataset_3km =
"IMD_BND_ESProtocol_Pacific_Guam_3km"
PacificSamoa_FeatureDataset_3km =
"IMD_BND_ESProtocol_Pacific_Samoa_3km"
USVI_FeatureDataset_3km = "IMD_BND_ESProtocol_USVI_3km"
Conus_FeatureDataset_30km = "IMD_BND_ESProtocol_CONUS_30km"
AK_FeatureDataset_30km = "IMD_BND_ESProtocol_Alaska_30km"
Hawaii_FeatureDataset_30km = "IMD_BND_ESProtocol_Hawaii_30km"
PacificSaipan_FeatureDataset_30km =
"IMD_BND_ESProtocol_Pacific_Saipan_30km"
PacificGuam_FeatureDataset_30km =
"IMD_BND_ESProtocol_Pacific_Guam_30km"
PacificSamoa_FeatureDataset_30km =
"IMD_BND_ESProtocol_Pacific_Samoa_30km"
USVI_FeatureDataset_30km = "IMD_BND_ESProtocol_USVI_30km_py"

Service_FeatureClass = "LandscapeDynamics_Dissolved"
AK_FeatureClass = "NPS_Alaska_Dissolved"
Hawaii_FeatureClass = "NPS_Hawaii_Dissolved"
PacificSaipan_FeatureClass = "NPS_Pacific_Saipan_Dissolved"
PacificGuam_FeatureClass = "NPS_Pacific_Guam_Dissolved"
PacificSamoa_FeatureClass = "NPS_Pacific_Samoa_Dissolved"
Conus_FeatureClass = "NPS_CONUS_Dissolved"
USVI_FeatureClass = "NPS_USVI_Dissolved"

# AllUnits array is populated from LDParks table in SQL
AllUnits = []
cursor = arcpy.SearchCursor(parameters[3].valueAsText)
for row in cursor:
    AllUnits.append(row.getValue("Code"))

NonConusUnits =
["ALAG", "ALKA", "AMME", "ANIA", "BELA", "BUIS", "CAKR", "DENA", "GAAR", "GLBA", "HA
LE", "HONO", "KALA", "KAHO", "KATM", "KEFJ", "KOVA",

"KLGO", "HAVO", "LACL", "NOAT", "NPSA", "PUHE", "PUHO", "SITK", "VALR", "VIIS", "YUC
H", "WAPA", "WRST", "FOMO"]

ServiceParks = AllUnits
arcpy.AddMessage(ServiceParks)
ConusUnits = list(set(AllUnits).difference(set(NonConusUnits)))

```

```

# if new parks (LD/IMD-related only) are added for these
geographic regions - these lists (and NonConusUnits) must be updated in
script.

Alaska_Parks =
"'ALAG','ANIA','BELA','CAKR','DENA','GAAR','GLBA','KATM','KEFJ','KOVA','KL
GO','LACL','NOAT','SITK','YUCH','WRST'"
Alaska_Parks_Array =
['ALAG','ANIA','BELA','CAKR','DENA','GAAR','GLBA','KATM','KEFJ','KOVA','KL
GO','LACL','NOAT','SITK','YUCH','WRST']
Hawaii_Parks =
"'ALKA','HALE','KALA','KAHO','HAVO','HONO','PUHE','PUHO','VALR'"
Hawaii_Parks_Array =
['ALKA','HALE','KALA','KAHO','HAVO','HONO','PUHE','PUHO','VALR']
PacificSaipan_Parks = "'AMME'"
PacificSaipan_Parks_Array = ['AMME']
PacificGuam_Parks = "'WAPA'"
PacificGuam_Parks_Array = ['WAPA']
PacificSamoa_Parks = "'NPSA'"
PacificSamoa_Parks_Array = ['NPSA']
USVI_Parks = "'BUIS','VIIS'"
USVI_Parks_Array = ['BUIS','VIIS']

Output_Layer = "AOABounds_Layer"
Output_Select = "AOABounds_Select"
OutputService = "LandscapeDynamics_Dissolved"

OutputsToDelete = [Output_Layer, Output_Select]

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

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

arcpy.CreateFileGDB_management(Output_Workspace,
"IMD_BND_ESProtocol_Park3km30km_AOA.gdb"); messages.addGPMessages()

# Copy LD boundary feature class into new file geodatabases
arcpy.MakeFeatureLayer_management(Source_FeatureClass,
Output_Layer,"Lifecycle = 'Active'"); 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')

        # Dissolve NPS boundary feature class on UNIT_CODE
        arcpy.MakeFeatureLayer_management(Output_TargetGDB + "\\\" +
Output_FeatureClass_Root, Output_Layer); messages.addGPMessages()
        # Dissolve acts as another check point
        arcpy.Dissolve_management(Output_Layer,Output_TargetGDB + "\\\" +
Dissolved_FeatureClass_Root,["Unit_Code_Display","UNIT_CODE","Lifecycle"])
; messages.addGPMessages()
        arcpy.RepairGeometry_management(Output_TargetGDB + "\\\" +
Dissolved_FeatureClass_Root); messages.addGPMessages()
        arcpy.MetadataImporter_conversion(Metadata_Template,
Output_TargetGDB + "\\\" + Dissolved_FeatureClass_Root);
arcpy.AddMessage('\n Created GDB and source feature class \n')

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

arcpy.CreateFeatureDataset_management(Output_TargetGDB,Conus_FeatureDatase
t_30km,ConusSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,AK_FeatureDataset_3
0km,AKSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,Hawaii_FeatureDatase
t_30km,HawaiiSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,PacificSaipan_Featu
reDataset_30km,PacificSaipanSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,PacificGuam_Feature
Dataset_30km,PacificSaipanSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,PacificSamoa_Featur
eDataset_30km,PacificSamoaSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,USVI_FeatureDataset
_30km,USVISpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,Conus_FeatureDatase
t_3km,ConusSpatialRef); messages.addGPMessages()

```

```

arcpy.CreateFeatureDataset_management(Output_TargetGDB,AK_FeatureDataset_3
km,AKSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,Hawaii_FeatureDatas
et_3km,HawaiiSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,PacificSaipan_Featu
reDataset_3km,PacificSaipanSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,PacificGuam_Feature
Dataset_3km,PacificSaipanSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,PacificSamoa_Featur
eDataset_3km,PacificSamoaSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,USVI_FeatureDataset
_3km,USVISpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,Conus_FeatureDatase
t_Park,ConusSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,AK_FeatureDataset_P
ark,AKSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,Hawaii_FeatureDatas
et_Park,HawaiiSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,PacificSaipan_Featu
reDataset_Park,PacificSaipanSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,PacificGuam_Feature
Dataset_Park,PacificSaipanSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,PacificSamoa_Featur
eDataset_Park,PacificSamoaSpatialRef); messages.addGPMessages()

arcpy.CreateFeatureDataset_management(Output_TargetGDB,USVI_FeatureDataset
_Park,USVISpatialRef); messages.addGPMessages()

# Populate the Source Feature Dataset - albers
albers = Output_TargetGDB + "\\\" + Dissolved_FeatureClass_Root
albers_park = Output_TargetGDB + "\\\" +
'IMD_BND_ESProtocol_Park_albers_py'
arcpy.MakeFeatureLayer_management(albers, Output_Select);
messages.addGPMessages()
arcpy.CopyFeatures_management(Output_Select, albers_park);
messages.addGPMessages()

```

```

        arcpy.RepairGeometry_management(albers_park);
messages.addGPMessages()
        arcpy.CopyRows_management(Unit_Lookups, Lookup);
messages.addGPMessages()

arcpy.JoinField_management(albers_park,"UNIT_CODE",Lookup,"PARK_CODE",["UNIT_NAME", "NETWORK_CODE", "NETWORK_NAME", "REGION_NAME", "PARK_CODE"]);
messages.addGPMessages()

arcpy.JoinField_management(albers_park,"UNIT_CODE",Output_Layer,"UNIT_CODE",["DATE_EDIT",
"Identifier","Feature_Source","Source_Metadata","DSLink","RedactReason","ChangeReason","CreateDate","UnitCodeQualifier","FeatureQualifier","DataCertificationLevel"]); messages.addGPMessages() #Source_FeatureClass
        FieldCalc = ("!UNIT_CODE!+_'+"+Output_FeatureClass_Park+"'")
        arcpy.AddField_management(albers_park, "AOA_NAME", "TEXT","","",
"50", "AOA_NAME","NULLABLE","REQUIRED",""); arcpy.AddMessage('\n Add Field AOA_NAME Park Service \n')
        arcpy.CalculateField_management(albers_park, "AOA_NAME",
FieldCalc,"PYTHON_9.3"); arcpy.AddMessage('\n Calculate Field AOA_NAME Park Service \n')

arcpy.CalculateField_management(albers_park,"UnitCodeQualifier","SC","PYTHON"); messages.addGPMessages()

arcpy.CalculateField_management(albers_park,"FeatureQualifier","NC","PYTHON"); messages.addGPMessages()

# Populate CONUS, AK, HI, Saipan, Guam, Samoa, USVI Feature
Classes
        arcpy.SelectLayerByAttribute_management(Output_Select,
"NEW_SELECTION", "UNIT_CODE IN (" +Alaska_Parks+)");
messages.addGPMessages()
        arcpy.SelectLayerByAttribute_management(Output_Select,
"REMOVE_FROM_SELECTION", "Lifecycle = 'Inactive'");
messages.addGPMessages()
        arcpy.env.outputCoordinateSystem = AKSpatialRef
        AK_ID = '"IMD-LDAOAPark-3338-"!Unit_Code_Display! "-"!CreateDate!'
        CreateServiceAOAs.createFeatureClass(self, Output_Select,
Output_TargetGDB, AK_FeatureClass, Unit_Lookups, Lookup, Status_Lookups,
Metadata_Template,AK_ID)
        arcpy.ClearEnvironment("outputCoordinateSystem")

        arcpy.SelectLayerByAttribute_management(Output_Select,
"NEW_SELECTION", "UNIT_CODE IN (" +Hawaii_Parks+)");
messages.addGPMessages()
        arcpy.SelectLayerByAttribute_management(Output_Select,
"REMOVE_FROM_SELECTION", "Lifecycle = 'Inactive'");
messages.addGPMessages()

```

```

        arcpy.env.outputCoordinateSystem = HawaiiSpatialRef
        HI_ID = '"IMD-LDAOAPark-26905-"!Unit_Code_Display!"-"!CreateDate!'
        CreateServiceAOAs.createFeatureClass(self, Output_Select,
        Output_TargetGDB, Hawaii_FeatureClass, Unit_Lookups, Lookup,
        Status_Lookups, Metadata_Template, HI_ID)
        arcpy.ClearEnvironment("outputCoordinateSystem")

        arcpy.SelectLayerByAttribute_management(Output_Select,
        "NEW_SELECTION", "UNIT_CODE IN (" + PacificSamoa_Parks + ")");
        messages.addGPMessages()
        arcpy.SelectLayerByAttribute_management(Output_Select,
        "REMOVE_FROM_SELECTION", "Lifecycle = 'Inactive'");
        messages.addGPMessages()
        arcpy.env.outputCoordinateSystem = PacificSamoaSpatialRef
        Sam_ID = '"IMD-LDAOAPark-2195-"!Unit_Code_Display!"-"!CreateDate!'
        CreateServiceAOAs.createFeatureClass(self, Output_Select,
        Output_TargetGDB, PacificSamoa_FeatureClass, Unit_Lookups, Lookup,
        Status_Lookups, Metadata_Template, Sam_ID)
        arcpy.ClearEnvironment("outputCoordinateSystem")

        arcpy.SelectLayerByAttribute_management(Output_Select,
        "NEW_SELECTION", "UNIT_CODE IN (" + USVI_Parks + ")");
        messages.addGPMessages()
        arcpy.SelectLayerByAttribute_management(Output_Select,
        "REMOVE_FROM_SELECTION", "Lifecycle = 'Inactive'");
        messages.addGPMessages()
        arcpy.env.outputCoordinateSystem = USVISpatialRef
        CreateServiceAOAs.createFeatureClass(self, Output_Select,
        Output_TargetGDB, USVI_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 (" + PacificGuam_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,
"ADD_TO_SELECTION", "UNIT_CODE IN (" + USVI_Parks + ")");
messages.addGPMessages()
        arcpy.SelectLayerByAttribute_management(Output_Select,
"SWITCH_SELECTION"); messages.addGPMessages()
        arcpy.SelectLayerByAttribute_management(Output_Select,
"REMOVE_FROM_SELECTION", "Lifecycle = 'Inactive'");
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.SelectLayerByAttribute_management(Output_Select,
"REMOVE_FROM_SELECTION", "Lifecycle = 'Inactive'");
messages.addGPMessages()
        arcpy.env.outputCoordinateSystem = PacificSaipanSpatialRef
        arcpy.env.geographicTransformations = (transformMethodSaipan);
messages.addGPMessages()
        SAIP_Guam_ID = '"IMD-LDAOAPark-32655-"'!Unit_Code_Display!"-
"!CreateDate!'
        CreateServiceAOAs.createFeatureClass(self, Output_Select,
Output_TargetGDB, PacificSaipan_FeatureClass, Unit_Lookups, Lookup,
Status_Lookups, Metadata_Template, SAIP_Guam_ID)

        arcpy.SelectLayerByAttribute_management(Output_Select,
"NEW_SELECTION", "UNIT_CODE IN (" + PacificGuam_Parks + ")");
messages.addGPMessages()
        arcpy.SelectLayerByAttribute_management(Output_Select,
"REMOVE_FROM_SELECTION", "Lifecycle = 'Inactive'");
messages.addGPMessages()
        CreateServiceAOAs.createFeatureClass(self, Output_Select,
Output_TargetGDB, PacificGuam_FeatureClass, Unit_Lookups, Lookup,
Status_Lookups, Metadata_Template, SAIP_Guam_ID)
        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 == 'IMD_BND_ESProtocol_CONUS_Park':
        park_List = ConusUnits
        park_fc = Conus_FeatureClass
        fd30 = Conus_FeatureDataset_30km
        fd3 = Conus_FeatureDataset_3km
    if fd == 'IMD_BND_ESProtocol_Alaska_Park':
        park_List = Alaska_Parks_Array
        park_fc = AK_FeatureClass
        fd30 = AK_FeatureDataset_30km
        fd3 = AK_FeatureDataset_3km
    elif fd == 'IMD_BND_ESProtocol_Hawaii_Park':
        park_List = Hawaii_Parks_Array
        park_fc = Hawaii_FeatureClass
        fd30 = Hawaii_FeatureDataset_30km
        fd3 = Hawaii_FeatureDataset_3km
    elif fd == 'IMD_BND_ESProtocol_Pacific_Saipan_Park':
        park_List = PacificSaipan_Parks_Array
        park_fc = PacificSaipan_FeatureClass
        fd30 = PacificSaipan_FeatureDataset_30km
        fd3 = PacificSaipan_FeatureDataset_3km
    elif fd == 'IMD_BND_ESProtocol_Pacific_Guam_Park':
        park_List = PacificGuam_Parks_Array
        park_fc = PacificGuam_FeatureClass
        fd30 = PacificGuam_FeatureDataset_30km
        fd3 = PacificGuam_FeatureDataset_3km
    elif fd == 'IMD_BND_ESProtocol_Pacific_Samoa_Park':
        park_List = PacificSamoa_Parks_Array
        park_fc = PacificSamoa_FeatureClass
        fd30 = PacificSamoa_FeatureDataset_30km
        fd3 = PacificSamoa_FeatureDataset_3km
    elif fd == 'IMD_BND_ESProtocol_USVI_Park':
        park_List = USVI_Parks_Array
        park_fc = USVI_FeatureClass
        fd30 = USVI_FeatureDataset_30km
        fd3 = USVI_FeatureDataset_3km
    elif fd == 'IMD_BND_ESProtocol_ServiceSourcePark':
        park_List = ServiceParks
        park_fc = OutputService
    else:
        pass

    for park in park_List:
        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", "UNI
T_CODE = '\"'+park+'\""); messages.addGPMessages()

arcpy.CopyFeatures_management(Output_Select, Output_TargetGDB + "\\\" + fd +
"\\\" + park_FeatureClass); 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 != 'IMD_BND_ESProtocol_ServiceSourcePark':

CreateServiceAOAs.populateFeatureClass(self, Output_Select, Output_TargetGDB
, fd30, buffered_FeatureClass, BufferDistance, Unit_Lookups)

CreateServiceAOAs.populateFeatureClass(self, Output_Select, Output_TargetGDB
, fd3, 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.SelectLayerByAttribute_management(Output_Select,
"CLEAR_SELECTION", ""); messages.addGPMessages()

        # Create complete (merged) feature class of AOAs for use in Stats
Tools
        Alaska_3km = Output_TargetGDB +
"\\\"+AK_FeatureDataset_3km+\"\\\"IMD_BND_ESProtocol_Alaska_\" +
Output_FeatureClass_3km
        Alaska_30km = Output_TargetGDB +
"\\\"+AK_FeatureDataset_30km+\"\\\"IMD_BND_ESProtocol_Alaska_\" +
Output_FeatureClass_30km
        CONUS_3km = Output_TargetGDB +
"\\\"+Conus_FeatureDataset_3km+\"\\\"IMD_BND_ESProtocol_CONUS_\" +
Output_FeatureClass_3km

```

```

        CONUS_30km = Output_TargetGDB +
        "\\ "+Conus_FeatureDataset_30km+"\\IMD_BND_ESProtocol_CONUS_" +
        Output_FeatureClass_30km
        Hawaii_3km = Output_TargetGDB +
        "\\ "+Hawaii_FeatureDataset_3km+"\\IMD_BND_ESProtocol_Hawaii_" +
        Output_FeatureClass_3km
        Hawaii_30km = Output_TargetGDB +
        "\\ "+Hawaii_FeatureDataset_30km+"\\IMD_BND_ESProtocol_Hawaii_" +
        Output_FeatureClass_30km
        Saipan_3km = Output_TargetGDB +
        "\\ "+PacificSaipan_FeatureDataset_3km+"\\IMD_BND_ESProtocol_Saipan_" +
        Output_FeatureClass_3km
        Saipan_30km = Output_TargetGDB +
        "\\ "+PacificSaipan_FeatureDataset_30km+"\\IMD_BND_ESProtocol_Saipan_" +
        Output_FeatureClass_30km
        Guam_3km = Output_TargetGDB +
        "\\ "+PacificGuam_FeatureDataset_3km+"\\IMD_BND_ESProtocol_Guam_" +
        Output_FeatureClass_3km
        Guam_30km = Output_TargetGDB +
        "\\ "+PacificGuam_FeatureDataset_30km+"\\IMD_BND_ESProtocol_Guam_" +
        Output_FeatureClass_30km
        Samoa_3km = Output_TargetGDB +
        "\\ "+PacificSamoa_FeatureDataset_3km+"\\IMD_BND_ESProtocol_Samoa_" +
        Output_FeatureClass_3km
        Samoa_30km = Output_TargetGDB +
        "\\ "+PacificSamoa_FeatureDataset_30km+"\\IMD_BND_ESProtocol_Samoa_" +
        Output_FeatureClass_30km
        USVI_3km = Output_TargetGDB +
        "\\ "+USVI_FeatureDataset_3km+"\\IMD_BND_ESProtocol_USVI_" +
        Output_FeatureClass_3km
        USVI_30km = Output_TargetGDB +
        "\\ "+USVI_FeatureDataset_30km+"\\IMD_BND_ESProtocol_USVI_" +
        Output_FeatureClass_30km

        CreateServiceAOAs.createBuffer(self,
        Output_TargetGDB,AK_FeatureClass,AK_FeatureDataset_Park,BufferDistance_3km
        ,BufferDistance,Alaska_3km,Alaska_30km,"Alaska_AOA_Park")
        CreateServiceAOAs.createBuffer(self,
        Output_TargetGDB,Conus_FeatureClass,Conus_FeatureDataset_Park,BufferDistan
        ce_3km,BufferDistance,CONUS_3km,CONUS_30km,"CONUS_AOA_Park")
        CreateServiceAOAs.createBuffer(self,
        Output_TargetGDB,Hawaii_FeatureClass,Hawaii_FeatureDataset_Park,BufferDist
        ance_3km,BufferDistance,Hawaii_3km,Hawaii_30km,"Hawaii_AOA_Park")
        CreateServiceAOAs.createBuffer(self,
        Output_TargetGDB,PacificSaipan_FeatureClass,PacificSaipan_FeatureDataset_P
        ark,BufferDistance_3km,BufferDistance,Saipan_3km,Saipan_30km,"Saipan_AOA_P
        ark")

```

```

        CreateServiceAOAs.createBuffer(self,
Output_TargetGDB,PacificGuam_FeatureClass,PacificGuam_FeatureDataset_Park,
BufferDistance_3km,BufferDistance,Guam_3km,Guam_30km,"Guam_AOA_Park")
        CreateServiceAOAs.createBuffer(self,
Output_TargetGDB,PacificSamoa_FeatureClass,PacificSamoa_FeatureDataset_Par
k,BufferDistance_3km,BufferDistance,Samoa_3km,Samoa_30km,"Samoa_AOA_Park")
        CreateServiceAOAs.createBuffer(self,
Output_TargetGDB,USVI_FeatureClass,USVI_FeatureDataset_Park,BufferDistance
_3km,BufferDistance,USVI_3km,USVI_30km,"USVI_AOA_Park")

        Albers3km_Dissolve = Output_TargetGDB + "\\\" +
"IMD_BND_ESProtocol_3km_albers_py"
        Albers30km_Dissolve = Output_TargetGDB + "\\\" +
"IMD_BND_ESProtocol_30km_albers_py"

arcpy.Buffer_analysis(albers_park,Albers3km_Dissolve,BufferDistance_3km,"F
ULL","ROUND","NONE"); messages.addMessage('\n Created albers dissolved
feature classes 3km \n')
        arcpy.CalculateField_management(Albers3km_Dissolve, "AOA_NAME",
"!UNIT_CODE!+'_AOA_3km'", "PYTHON_9.3"); arcpy.AddMessage('\n Calculate
Field AOA_NAME 3km Service \n')

arcpy.CalculateField_management(Albers3km_Dissolve,"Identifier","IMD-
LDAOA3km-102039-\"!Unit_Code_Display!\"-\"!CreateDate!\",\"PYTHON\");
messages.addGPMessages()

arcpy.Buffer_analysis(albers_park,Albers30km_Dissolve,BufferDistance,"FULL
","ROUND","NONE"); arcpy.AddMessage('\n Created albers dissolved feature
classes 30km \n')
        arcpy.CalculateField_management(Albers30km_Dissolve, "AOA_NAME",
"!UNIT_CODE!+'_AOA_30km'", "PYTHON_9.3"); arcpy.AddMessage('\n Calculate
Field AOA_NAME 30km Service \n')

arcpy.CalculateField_management(Albers30km_Dissolve,"Identifier","IMD-
LDAOA30km-102039-\"!Unit_Code_Display!\"-\"!CreateDate!\",\"PYTHON\");
messages.addGPMessages()

        WGS30km = Output_TargetGDB + "\\\" +
"IMD_BND_ESProtocol_30km_wgs_py"
        WGS3km = Output_TargetGDB + "\\\"+ "IMD_BND_ESProtocol_3km_wgs_py"
        WGS = Output_TargetGDB + "\\\"+ "IMD_BND_ESProtocol_Park_wgs_py"

arcpy.Project_management(Albers30km_Dissolve,WGS30km,WGSSpatialRef,transfo
rmMethodSaipan); messages.addGPMessages()

arcpy.Project_management(Albers3km_Dissolve,WGS3km,WGSSpatialRef,transform
MethodSaipan); messages.addGPMessages()

```

```

arcpy.Project_management(albers_park,WGS,WGSSpatialRef,transformMethodSaip
an); messages.addGPMessages()
    arcpy.RepairGeometry_management(WGS); messages.addGPMessages()
    arcpy.RepairGeometry_management(WGS3km); messages.addGPMessages()
    arcpy.RepairGeometry_management(WGS30km); messages.addGPMessages()
    arcpy.CalculateField_management(WGS,"Identifier","IMD-LDAOAPark-
4326-"!Unit_Code_Display!"-"!CreateDate!',"PYTHON");
messages.addGPMessages()
    arcpy.CalculateField_management(WGS30km,"Identifier","IMD-
LDAOA30km-4326-"!Unit_Code_Display!"-"!CreateDate!',"PYTHON");
messages.addGPMessages()
    arcpy.CalculateField_management(WGS3km,"Identifier","IMD-
LDAOA3km-4326-"!Unit_Code_Display!"-"!CreateDate!',"PYTHON");
messages.addGPMessages()

```

Appendix D. Additional Processing Script for Standard AOAs

In a stand-alone IDE that supports python, run the following two additional scripts to produce the output dataset. Run them separately and in the order presented.

Script #1

```
import arcpy
import os, time

arcpy.env.overwriteOutput = 1

arcpy.env.workspace = r'' # this value is the file geodatabase of the
standard AOAs

gdbToBeUpdated = r'' # this value should match arcpy.env.workspace

DomainGDB = r'' # Domain GDB: see cited reference (NPS IMD 2018d)

datasets = arcpy.ListDatasets(feature_type='feature')
datasets = datasets if datasets is not None else []
arcpy.AddMessage("\n Datasets: "+str(datasets))

pyToBeUpdated =
["_30km_albers_py", "_3km_albers_py", "_Park_albers_py", "_30km_wgs_py", "_3km
_wgs_py", "_Park_wgs_py"]

DOM_Change = 'DOM_ChangeReasonFromPrecedingVersion_NPSIMD2019'
DOM_DataCert = 'DOM_DataCertificationLevel_NPSIMD2019'
DOM_EDITED = 'DOM_EDITEDBY_NPSIMD2019'
DOM_FeatAlt = 'DOM_FeatureAlteration_NPSIMD2019'
DOM_FeatLife = 'DOM_FeatureLifecycle_NPSIMD2019'
DOM_ORIG = 'DOM_ORIGINATOR_NPSIMD2019'
DOM_UnitCodeAlt = 'DOM_UnitCodeAlteration_NPSIMD2019'

for ds in datasets:
    for fc in arcpy.ListFeatureClasses(feature_dataset=ds):
        path = os.path.join(gdbToBeUpdated, ds, fc)
        arcpy.AddMessage("\n Remove Domains from "+path)

arcpy.RemoveDomainFromField_management(path, "DataCertificationLevel")
arcpy.RemoveDomainFromField_management(path, "FeatureQualifier")
arcpy.RemoveDomainFromField_management(path, "Feature_Source")
arcpy.RemoveDomainFromField_management(path, "Lifecycle")
arcpy.RemoveDomainFromField_management(path, "UnitCodeQualifier")
arcpy.RemoveDomainFromField_management(path, "ChangeReason")
arcpy.RemoveDomainFromField_management(path, "RedactReason")

for py in pyToBeUpdated:
```

```

path = os.path.join(gdbToBeUpdated, "IMD_BND_ESProtocol"+py)
arcpy.AddMessage("\n Remove Domains from "+path)
arcpy.RemoveDomainFromField_management(path, "DataCertificationLevel")
arcpy.RemoveDomainFromField_management(path, "FeatureQualifier")
arcpy.RemoveDomainFromField_management(path, "Feature_Source")
arcpy.RemoveDomainFromField_management(path, "UnitCodeQualifier")
arcpy.RemoveDomainFromField_management(path, "ChangeReason")
arcpy.RemoveDomainFromField_management(path, "RedactReason")

arcpy.AddMessage("\n Delete Domains from "+gdbToBeUpdated)
arcpy.DeleteDomain_management(gdbToBeUpdated, "ChangeReason_Domain")
arcpy.DeleteDomain_management(gdbToBeUpdated, "DataCertificationLevel_Domain")
arcpy.DeleteDomain_management(gdbToBeUpdated, "FeatureQualifier_Domain")
arcpy.DeleteDomain_management(gdbToBeUpdated, "Feature_Source")
arcpy.DeleteDomain_management(gdbToBeUpdated, "RedactReason_Domain")
arcpy.DeleteDomain_management(gdbToBeUpdated, "UnitCodeQualifier_Domain")

for ds in datasets:
    for fc in arcpy.ListFeatureClasses(feature_dataset=ds):
        path = os.path.join(gdbToBeUpdated, ds, fc)

arcpy.AlterField_management(path, "UNIT_CODE", "UNITCODE", 'UNITCODE', "TEXT",
16, "NULLABLE", 'None')
    arcpy.AddMessage('\n Alter Field UNIT_CODE to UNITCODE:
'+str(path)+'\n')

arcpy.AlterField_management(path, "Full_Name", "UNITNAME", "UNITNAME", "TEXT",
254, "NULLABLE", 'None')
    arcpy.AddMessage('\n Alter Field Full_Name to UNITNAME:
'+str(path)+'\n')

arcpy.AlterField_management(path, "DATE_EDIT", "EDITDATE", "EDITDATE", "DATE",
"", "NULLABLE", "None")
    arcpy.AddMessage('\n Alter Field DATE_EDIT to EDITDATE:
'+str(path)+'\n')

arcpy.AlterField_management(path, "Unit_Code_Display", "UnitCodeDisplay", "UnitCodeDisplay", "TEXT", 25, "NULLABLE", 'None')
    arcpy.AddMessage('\n Alter Fields Unit_Code_Display to
UnitCodeDisplay '+str(path)+'\n')

arcpy.AlterField_management(path, "Feature_Source", "ORIGINATOR", "ORIGINATOR", "TEXT", 100, "NULLABLE", 'None')
    arcpy.AddMessage('\n Alter Field Feature_Source to ORIGINATOR:
'+str(path)+'\n')

```

```

arcpy.AlterField_management(path,"Source_Metadata","OriginatorMetadata","OriginatorMetadata","TEXT",150,"NULLABLE",'None')
    arcpy.AddMessage('\n Alter Field Source_Metadata to OriginatorMetadata '+str(path)+'\n')

arcpy.AlterField_management(path,"Identifier","GEOMETRYID","GEOMETRYID","TEXT",250,"NULLABLE",'None')
    arcpy.AddMessage('\n Alter Field Identifier to GEOMETRYID: '+str(path)+'\n')

arcpy.AlterField_management(path,"CreateDate","IMDCreateDate","IMDCreateDate","TEXT",10,"NULLABLE",'None')
    arcpy.AddMessage('\n Alter Field CreateDate to IMDCreateDate: '+str(path)+'\n')

arcpy.AlterField_management(path,"PARK_CODE","ParkCode","ParkCode","TEXT",4,"NULLABLE",'None')
    arcpy.AddMessage('\n Alter Field PARK_CODE to ParkCode: '+str(path)+'\n')

    fields = [f.name for f in arcpy.ListFields(path)]
    if "AREA_SQKM" in fields:
        arcpy.AddMessage('\n Add/Calculate Field AREASQKM: '+str(path)+'\n')
        arcpy.DeleteField_management(path,"BUFF_DIST")

arcpy.AddField_management(path,"AREASQKM","FLOAT","", "", "", "AREASQKM", "NULLABLE", "NON_REQUIRED")

arcpy.CalculateField_management(path,"AREASQKM", '!AREA_SQKM!', "PYTHON_9.3")
    arcpy.DeleteField_management(path,"AREA_SQKM")

arcpy.AlterField_management(path,"DSLink","FeatureMetadata","FeatureMetadata","TEXT",150,"NULLABLE",'None')
    arcpy.AddMessage('\n Alter Field DSLink to FeatureMetadata: '+str(path)+'\n')

arcpy.AlterField_management(path,"Lifecycle","FeatureLifecycle","FeatureLifecycle","TEXT",15,"NULLABLE",'None')
    arcpy.AddMessage('\n Alter Field Lifecycle to FeatureLifecycle: '+str(path)+'\n')

arcpy.AddField_management(path,"ChangeReasonFromPrecedingVersion","TEXT", '', '', 50, "ChangeReasonFromPrecedingVersion", "NULLABLE", "NON_REQUIRED")

```

```

arcpy.CalculateField_management(path,"ChangeReasonFromPrecedingVersion",'!
ChangeReason!',"PYTHON_9.3")
    arcpy.AddMessage('\n Add Field/Calculate Field
ChangeReasonFromPrecedingVersion: '+str(path)+'\n')
    arcpy.DeleteField_management(path,"ChangeReason")
    arcpy.DeleteField_management(path,"RedactReason")
    arcpy.AddMessage('\n Delete Fields ChangeReason and RedactReason:
'+str(path)+'\n')

arcpy.AddField_management(path,"EDITEDBY","TEXT","",100,"EDITEDBY","NUL
LABLE","NON_REQUIRED")
    arcpy.AddMessage('\n Add Field EDITEDBY: '+str(path)+'\n')

arcpy.AlterField_management(path,"UnitCodeQualifier","UnitCodeAlteration",
"UnitCodeAlteration","TEXT",10,'NULLABLE','None')
    arcpy.AddMessage('\n Alter Field UnitCodeQualifier to
UnitCodeAlteration: '+str(path)+'\n')

arcpy.AlterField_management(path,"FeatureQualifier","FeatureAlteration","F
eatureAlteration","TEXT",10,"NULLABLE","NON_REQUIRED")
    arcpy.AddMessage('\n Alter Field FeatureQualifier to
FeatureAlteration: '+str(path)+'\n')

    if 'GRTE' in fc:

arcpy.CalculateField_management(path,"FeatureAlteration","NC","PYTHON_9.
3")

arcpy.JoinField_management(path,"UnitCodeAlteration",os.path.join(DomainGD
B,DOM_UnitCodeAlt),"Code","UnitCodeAlteration_Label")
    arcpy.AddMessage('\n Join Field UnitCodeAlteration_Label:
'+str(path)+'\n')

arcpy.JoinField_management(path,"FeatureAlteration",os.path.join(DomainGDB
,DOM_FeatAlt),"Code","FeatureAlteration_Label")
    arcpy.AddMessage('\n Join Field FeatureAlteration_Label:
'+str(path)+'\n')

    arcpy.MakeFeatureLayer_management(path,"py_temp")
    arcpy.AddMessage('\n Make Feature Layer: '+str(path)+'\n')

arcpy.SelectLayerByAttribute_management("py_temp","NEW_SELECTION","ChangeR
easonFromPrecedingVersion = 'B'")

arcpy.CalculateField_management("py_temp","ChangeReasonFromPrecedingVersio
n","LRD","PYTHON_9.3")

```

```

arcpy.SelectLayerByAttribute_management("py_temp", "SWITCH_SELECTION")

arcpy.CalculateField_management("py_temp", "ChangeReasonFromPrecedingVersion", '"NC"', "PYTHON_9.3")
    arcpy.AddMessage('\n Calculate Field
ChangeReasonFromPrecedingVersion: '+str(path)+'\n')

arcpy.SelectLayerByAttribute_management("py_temp", "CLEAR_SELECTION")

arcpy.JoinField_management(path, "ChangeReasonFromPrecedingVersion", os.path
.join(DomainGDB, DOM_Change), 'Code', "ChangeReasonFromPrecedingVersion_Label")
    arcpy.AddMessage('\n Join Field
ChangeReasonFromPrecedingVersion_Label: '+str(path)+'\n')

arcpy.SelectLayerByAttribute_management("py_temp", "CLEAR_SELECTION")

    arcpy.CalculateField_management("py_temp", "ORIGINATOR", '"IMD-
Central Office"', "PYTHON_9.3")

arcpy.CalculateField_management("py_temp", "OriginatorMetadata", '"https://i
rma.nps.gov/DataStore/Reference/Profile/2237295"', "PYTHON_9.3")
    arcpy.AddMessage('\n Calculate Fields ORIGINATOR and
OriginatorMetadata: '+str(path)+'\n')

arcpy.SelectLayerByAttribute_management("py_temp", "CLEAR_SELECTION")

arcpy.CalculateField_management(path, "EDITEDBY", '"!ORIGINATOR!"', "PYTHON_9.3")
    arcpy.AddMessage('\n Calculate Field EDITEDBY: '+str(path)+'\n')
    arcpy.DeleteField_management(path, "ChangeReason_Label")
    arcpy.DeleteField_management(path, "FeatureQualifier_Label")
    arcpy.DeleteField_management(path, "UnitCodeQualifier_Label")
    arcpy.AddMessage('\n Delete Fields ChangeReason_Label and
FeatureQualifier_Label and UnitCodeQualifier_Label: '+str(path)+'\n')
    arcpy.AddMessage('\n DONE PROCESSING '+str(path)+'\n')

for py in pyToBeUpdated:
    path = os.path.join(gdbToBeUpdated, "IMD_BND_ESProtocol"+py)
    arcpy.AddMessage('\n Alter Fields: '+str(path)+'...')

arcpy.AlterField_management(path, "UNIT_CODE", "UNITCODE", 'UNITCODE', "TEXT",
16, "NULLABLE", 'None')

arcpy.AlterField_management(path, "UNIT_NAME", "UNITNAME", "UNITNAME", "TEXT",
254, "NULLABLE", 'None')

```

```

arcpy.AlterField_management(path,"DATE_EDIT","EDITDATE","EDITDATE","DATE",
"", "NULLABLE", "None")

arcpy.AlterField_management(path,"Unit_Code_Display","UnitCodeDisplay","Un
itCodeDisplay","TEXT",25,"NULLABLE",'None')

arcpy.AlterField_management(path,"Feature_Source","ORIGINATOR","ORIGINATOR
","TEXT",100,"NULLABLE",'None')

arcpy.AlterField_management(path,"Source_Metadata","OriginatorMetadata","O
riginatorMetadata","TEXT",150,"NULLABLE",'None')

arcpy.AlterField_management(path,"Identifier","GEOMETRYID","GEOMETRYID","T
EXT",250,"NULLABLE",'None')

arcpy.AddField_management(path,"AOANAME","TEXT","", "",25,"AOANAME","NULLAB
LE","NON_REQUIRED")

arcpy.CalculateField_management(path,"AOANAME",'!AOA_NAME!',"PYTHON_9.3")
##
arcpy.AlterField_management(path,"AOA_NAME","AOANAME","AOANAME","TEXT",50,
"NULLABLE",'None')

arcpy.AlterField_management(path,"CreateDate","IMDCreateDate","IMDCreateDa
te","TEXT",10,"NULLABLE",'None')

arcpy.AlterField_management(path,"REGION_NAME","REGIONNAME","REGIONNAME","
TEXT",100,"NULLABLE",'None')

arcpy.AlterField_management(path,"NETWORK_CODE","NETWORKCODE","NETWORKCODE
","TEXT",4,"NULLABLE",'None')

arcpy.AlterField_management(path,"NETWORK_NAME","NETWORKNAME","NETWORKNAME
","TEXT",100,"NULLABLE",'None')

arcpy.AlterField_management(path,"PARK_CODE","Park_Code","ParkCode","TEXT"
,4,"NULLABLE",'None')

arcpy.AlterField_management(path,"DSLink","FeatureMetadata","FeatureMetada
ta","TEXT",150,"NULLABLE",'None')

arcpy.AlterField_management(path,"Lifecycle","FeatureLifecycle","FeatureLi
fecycle","TEXT",15,"NULLABLE",'None')

```

```

arcpy.AddField_management(path,"ChangeReasonFromPrecedingVersion","TEXT",'
',' ',50,"ChangeReasonFromPrecedingVersion","NULLABLE","NON_REQUIRED")

arcpy.CalculateField_management(path,"ChangeReasonFromPrecedingVersion",'!
ChangeReason!',"PYTHON_9.3")
    arcpy.DeleteField_management(path,"ChangeReason")
    arcpy.DeleteField_management(path,"RedactReason")

arcpy.AddField_management(path,"EDITEDBY","TEXT",' ',' ',100,"EDITEDBY","NUL
LABLE","NON_REQUIRED")

arcpy.AlterField_management(path,"UnitCodeQualifier","UnitCodeAlteration",
"UnitCodeAlteration","TEXT",10,'NULLABLE','None')

arcpy.AlterField_management(path,"FeatureQualifier","FeatureAlteration","F
eatureAlteration","TEXT",10,"NULLABLE","NON_REQUIRED")

arcpy.JoinField_management(path,"UnitCodeAlteration",os.path.join(DomainGD
B,DOM_UnitCodeAlt),"Code","UnitCodeAlteration_Label")

arcpy.JoinField_management(path,"FeatureAlteration",os.path.join(DomainGDB
,DOM_FeatAlt),"Code","FeatureAlteration_Label")

    arcpy.MakeFeatureLayer_management(path,"py_temp")

arcpy.SelectLayerByAttribute_management("py_temp","NEW_SELECTION","ChangeR
easonFromPrecedingVersion = 'B'")

arcpy.CalculateField_management("py_temp","ChangeReasonFromPrecedingVersio
n","'LRD'", "PYTHON_9.3")
    arcpy.SelectLayerByAttribute_management("py_temp","SWITCH_SELECTION")

arcpy.CalculateField_management("py_temp","ChangeReasonFromPrecedingVersio
n","'NC'", "PYTHON_9.3")

    arcpy.SelectLayerByAttribute_management("py_temp","CLEAR_SELECTION")

arcpy.JoinField_management(path,"ChangeReasonFromPrecedingVersion",os.path
.join(DomainGDB,DOM_Change),'Code',"ChangeReasonFromPrecedingVersion_Label
")
    arcpy.SelectLayerByAttribute_management("py_temp","CLEAR_SELECTION")

    arcpy.CalculateField_management("py_temp","ORIGINATOR","'IMD-Central
Office'", "PYTHON_9.3")

```

```

arcpy.CalculateField_management("py_temp", "OriginatorMetadata", '"https://irma.nps.gov/DataStore/Reference/Profile/2237295"', "PYTHON_9.3")
    arcpy.SelectLayerByAttribute_management("py_temp", "CLEAR_SELECTION")

arcpy.CalculateField_management(path, "EDITEDBY", "!ORIGINATOR!", "PYTHON_9.3")
    arcpy.DeleteField_management(path, "ChangeReason_Label")
    arcpy.DeleteField_management(path, "FeatureQualifier_Label")
    arcpy.DeleteField_management(path, "UnitCodeQualifier_Label")
    ##arcpy.DeleteField_management(path, "AOA_NAME")

arcpy.AddMessage('\n Copy Rows to Table \n')
arcpy.CopyRows_management(os.path.join(DomainGDB, DOM_Change), os.path.join(gdbToBeUpdated, DOM_Change))
arcpy.CopyRows_management(os.path.join(DomainGDB, DOM_DataCert), os.path.join(gdbToBeUpdated, DOM_DataCert))
arcpy.CopyRows_management(os.path.join(DomainGDB, DOM_EDITED), os.path.join(gdbToBeUpdated, DOM_EDITED))
arcpy.CopyRows_management(os.path.join(DomainGDB, DOM_FeatAlt), os.path.join(gdbToBeUpdated, DOM_FeatAlt))
arcpy.CopyRows_management(os.path.join(DomainGDB, DOM_FeatLife), os.path.join(gdbToBeUpdated, DOM_FeatLife))
arcpy.CopyRows_management(os.path.join(DomainGDB, DOM_ORIG), os.path.join(gdbToBeUpdated, DOM_ORIG))
arcpy.CopyRows_management(os.path.join(DomainGDB, DOM_UnitCodeAlt), os.path.join(gdbToBeUpdated, DOM_UnitCodeAlt))

arcpy.AddMessage('\n Table to Domain \n')
arcpy.TableToDomain_management(os.path.join(gdbToBeUpdated, DOM_Change), 'Code', 'Code', gdbToBeUpdated, DOM_Change, "Reason for feature change compared to previous version of dataset.")
arcpy.TableToDomain_management(os.path.join(gdbToBeUpdated, DOM_DataCert), 'Code', 'Code', gdbToBeUpdated, DOM_DataCert, "Data Certification Status.")
arcpy.TableToDomain_management(os.path.join(gdbToBeUpdated, DOM_EDITED), 'Code', 'Label', gdbToBeUpdated, DOM_EDITED, "Who last edited the feature. May not be the same as ORIGINATOR.")
arcpy.TableToDomain_management(os.path.join(gdbToBeUpdated, DOM_FeatAlt), 'Code', 'Code', gdbToBeUpdated, DOM_FeatAlt, "How feature differs from NPS Land Resources Division dataset.")
arcpy.TableToDomain_management(os.path.join(gdbToBeUpdated, DOM_FeatLife), 'Code', 'Code', gdbToBeUpdated, DOM_FeatLife, "Feature status.")
arcpy.TableToDomain_management(os.path.join(gdbToBeUpdated, DOM_ORIG), 'Code', 'Label', gdbToBeUpdated, DOM_ORIG, "Who originally created the feature. May not be the same as EDITEDBY.")
arcpy.TableToDomain_management(os.path.join(gdbToBeUpdated, DOM_UnitCodeAlt), 'Code', 'Code', gdbToBeUpdated, DOM_UnitCodeAlt, "How unit code differs from NPS Land Resources Division dataset.")

```

```

for ds in datasets:
    for fc in arcpy.ListFeatureClasses(feature_dataset=ds):
        path = os.path.join(gdbToBeUpdated,ds,fc)
        arcpy.AddMessage('\n Assigning Domains to '+str(path)+'...')

arcpy.AssignDomainToField_management(path,"ChangeReasonFromPrecedingVersion",DOM_Change)

arcpy.AssignDomainToField_management(path,"DataCertificationLevel",DOM_DataCert)
    arcpy.AssignDomainToField_management(path,"EDITEDBY",DOM_EDITED)

arcpy.AssignDomainToField_management(path,"FeatureAlteration",DOM_FeatAlt)

arcpy.AssignDomainToField_management(path,"FeatureLifecycle",DOM_FeatLife)
    arcpy.AssignDomainToField_management(path,"ORIGINATOR",DOM_ORIG)

arcpy.AssignDomainToField_management(path,"UnitCodeAlteration",DOM_UnitCodeAlt)

    fields = [f.name for f in arcpy.ListFields(path)]
    if "ORIG_FID" in fields:
        arcpy.DeleteField_management(path,"BUFF_DIST")
        arcpy.DeleteField_management(path,"ORIG_FID")
        arcpy.AddMessage('\n Delete Fields BUFF_DIST and ORIG_FID:
'+str(path)+'\n')
    else:
        pass

for py in pyToBeUpdated:
    path = os.path.join(gdbToBeUpdated,"IMD_BND_ESProtocol"+py)
    arcpy.AddMessage('\n Assigning Domains to '+str(path)+'...')

arcpy.AssignDomainToField_management(path,"ChangeReasonFromPrecedingVersion",DOM_Change)

arcpy.AssignDomainToField_management(path,"DataCertificationLevel",DOM_DataCert)
    arcpy.AssignDomainToField_management(path,"EDITEDBY",DOM_EDITED)

arcpy.AssignDomainToField_management(path,"FeatureAlteration",DOM_FeatAlt)

arcpy.AssignDomainToField_management(path,"FeatureLifecycle",DOM_FeatLife)
    arcpy.AssignDomainToField_management(path,"ORIGINATOR",DOM_ORIG)

arcpy.AssignDomainToField_management(path,"UnitCodeAlteration",DOM_UnitCodeAlt)

```

```

fields = [f.name for f in arcpy.ListFields(path)]
if "ORIG_FID" in fields:
    arcpy.DeleteField_management(path, "BUFF_DIST")
    arcpy.DeleteField_management(path, "ORIG_FID")
    arcpy.AddMessage('\n Delete Fields BUFF_DIST and ORIG_FID:
'+str(path)+'\n')
else:
    pass

```

Script #2

```

import arcpy
import os, time
import arcpy
import os

arcpy.env.overwriteOutput = 1
arcpy.env.workspace = r'' # Output GDB from Script #1
outFolder = r'' # output location
newGDB = os.path.join(outFolder, "IMD_BND_Protocol")

datasets = arcpy.ListDatasets(feature_type='feature')
datasets = datasets if datasets is not None else []

for ds in datasets:
    for fc in arcpy.ListFeatureClasses(feature_dataset=ds):
        path = os.path.join(arcpy.env.workspace, ds, fc)
        arcpy.AddMessage('\n Adding AOAName, NETWORKNAME, NETWORKCODE, and
REGIONCODE to '+str(path)+'...')

arcpy.JoinField_management(path, "UNITCODE", os.path.join(arcpy.env.workspac
e, "IMD_BND_ESProtocol_30km_albers_py"), "UNITCODE", ["REGIONNAME", "NETWORKCO
DE", "NETWORKNAME"])

arcpy.AddField_management(path, "AOAName", "TEXT", "", "", 25, "", "NULLABLE", "NO
N_REQUIRED")
    if 'Park' in fc:
        arcpy.AddMessage('\n Calculate AOAName for '+str(fc)+'...')

arcpy.CalculateField_management(path, "AOAName", '!UNITCODE! "_AOA_Park"', "PYT
HON_9.3")
    elif '3km' in fc:
        arcpy.AddMessage('\n Calculate AOAName for '+str(fc)+'...')

arcpy.CalculateField_management(path, "AOAName", '!UNITCODE! "_AOA_3km"', "PYT
HON_9.3")
    elif '30km' in fc:
        arcpy.AddMessage('\n Calculate AOAName for '+str(fc)+'...')

```

```

arcpy.CalculateField_management(path, "AOAName", '!UNITCODE! "_AOA_30km"', "PYTHON_9.3")
    else:
        pass

pyToBeUpdated =
["_30km_albers_py", "_3km_albers_py", "_Park_albers_py", "_30km_wgs_py", "_3km_wgs_py", "_Park_wgs_py"]

for py in pyToBeUpdated:
    path = os.path.join(arcpy.env.workspace, "IMD_BND_ESProtocol"+py)
    arcpy.AddMessage('\n Editing fields for '+str(path)+'...')
    if "wgs" in py:
        pass
    else:
        arcpy.AddMessage('\n Adding AREASQKM to '+str(path)+'...')

arcpy.AddField_management(path, "AREASQKM", "FLOAT", "", "", "", "", "NULLABLE", "NON_REQUIRED")

arcpy.CalculateField_management(path, "AREASQKM", "!shape.area@squarekilometers!", "PYTHON_9.3")

    fields = [f.name for f in arcpy.ListFields(path)]
    if "AOA_NAME" in fields:
        arcpy.DeleteField_management(path, "AOA_NAME")
        arcpy.AddMessage('\n Delete Field AOA_NAME: '+str(path)+'\n')
    else:
        pass

for py in pyToBeUpdated:
    path = os.path.join(arcpy.env.workspace, "IMD_BND_ESProtocol"+py)
    arcpy.AddMessage('\n Editing fields for '+str(path)+'...')
    if
arcpy.Exists(os.path.join(outFolder, "IMD_BND_ESProtocol_Park3km30kmCombo_AOA.gdb")):

arcpy.CopyFeatures_management(path, os.path.join(newGDB, "IMD_BND_ESProtocol"+py))
    else:

arcpy.CreateFileGDB_management(outFolder, "IMD_BND_ESProtocol_Park3km30kmCombo_AOA.gdb")

arcpy.CopyFeatures_management(path, os.path.join(newGDB, "IMD_BND_ESProtocol"+py))

```

```

arcpy.CopyRows_management(os.path.join(arcpy.env.workspace,"DOM_ChangeReasonFromPrecedingVersion_NPSIMD2019"),os.path.join(newGDB,"DOM_ChangeReasonFromPrecedingVersion_NPSIMD2019"))
arcpy.CopyRows_management(os.path.join(arcpy.env.workspace,"DOM_DataCertificationLevel_NPSIMD2019"),os.path.join(newGDB,"DOM_DataCertificationLevel_NPSIMD2019"))
arcpy.CopyRows_management(os.path.join(arcpy.env.workspace,"DOM_EDITEDBY_NPSIMD2019"),os.path.join(newGDB,"DOM_EDITEDBY_NPSIMD2019"))
arcpy.CopyRows_management(os.path.join(arcpy.env.workspace,"DOM_FeatureAlteration_NPSIMD2019"),os.path.join(newGDB,"DOM_FeatureAlteration_NPSIMD2019"))
arcpy.CopyRows_management(os.path.join(arcpy.env.workspace,"DOM_FeatureLifecycle_NPSIMD2019"),os.path.join(newGDB,"DOM_FeatureLifecycle_NPSIMD2019"))
arcpy.CopyRows_management(os.path.join(arcpy.env.workspace,"DOM_ORIGINATOR_NPSIMD2019"),os.path.join(newGDB,"DOM_ORIGINATOR_NPSIMD2019"))
arcpy.CopyRows_management(os.path.join(arcpy.env.workspace,"DOM_UnitCodeAlteration_NPSIMD2019"),os.path.join(newGDB,"DOM_UnitCodeAlteration_NPSIMD2019"))

```

Appendix E. Session and Version Information

Windows 10 Enterprise
Version 1803
Installed on 9/26/201
OS build 17134.984
ArcMap/Catalog 10.6.1.9270
Python 2.7.3 64-bit
PyScripter 3.6.0.0 x64

The Department of the Interior protects and manages the nation's natural resources and cultural heritage; provides scientific and other information about those resources; and honors its special responsibilities to American Indians, Alaska Natives, and affiliated Island Communities.

NPS 999/167725, March 2020

National Park Service
U.S. Department of the Interior



[Natural Resource Stewardship and Science](#)

1201 Oakridge Drive, Suite 150
Fort Collins, CO 80525