EXPOSING AND PROVIDING ACCESS TO INDIAN BIORESOURCE INFORMATION NETWORK (IBIN) SPECIES OCCURRENCE DATASET AS WEB SERVICE USING OGC WPS STANDARD

Kapil Oberai [1, *], Mayank Jasoria [2], Sameer Saran [1]

[1]Indian Institute of Remote Sensing, ISRO, Dehradun, India – (kapil, sameer)@iirs.gov.in
[2]Birla Institute of Technology and Science, Pilani, India - f2016703@pilani.bits-pilani.ac.in

Commission V, SS: Utilization of Bio-Resource National Databases

KEY WORDS: IBIN, Species Occurrence, Web Service, Open Source, WPS, Zoo-project, Asynchronous, WMS

ABSTRACT:

Species occurrence data are collected by many researchers worldwide as record of species present at a specific time at some defined place as part of biological field investigation serving as primary or secondary dataset. These datasets reside in separate silos across numerous distributed systems having different formats limiting its usage to full potential. IBIN portal provides a single window for accessing myriad spatial/non-spatial data on bioresources of the country. To promote reuse of occurrence dataset among organizations in an interoperable format including support for integration across various platforms & programming languages, it is been exposed as web service using OGC Web Processing Service (WPS) standard. WPS provides standardized interface for performing online geo-processing by exposing spatial processes, algorithms and calculations thereby enabling machine to machine communication and wider usage in various scenarios (e.g. service chaining etc.). Open source ZOO-project is used for developing the ‘Species Search’ WPS service. WPS takes inputs as either the species name or bounding box or shapefile defining the area of interest and returns queryable OGC complaint Web Map Service (WMS) as output with specie(s) occurrences represented in grid (5km x 5km) format, with each grid possessing attributes like specie(s) name, family, state, medicinal detail etc. WPS process can be invoked asynchronously, enabling proper feedback regarding status of the job submitted. JavaScript based web client for consuming this service has also been developed along with custom QGIS plugin to allow potential users to access the same in GIS software for wider reusability.

1. INTRODUCTION

Species occurrence data has been collected for a long time only as physical specimens and stored in museums as natural history. Such data is collected by many researchers as it finds many applications in various fields like biogeographical studies, conservation planning, bioprospecting (Chapman, 2005), species distribution prediction (Elith et al., 2006), estimating magnitudes of animal movements (Stewart et al., 2018) etc. In recent times, however, museums and other agencies have spent considerable amounts to support the digitization of such data into online species occurrence databases (Ball-Damerow et al., 2017).

These databases are managed by different bodies, meaning that they reside in various distributed networks, and each such database has a different format for storage and retrieval of data. Further, the data collected is usually documented and organised in a extremely inconsistent and fragmented approach (Dubois et al., 2013). This creates a problem as separate procedures are required to gather the same data from different databases, thereby limiting the use of datasets from multiple databases to its full potential. The Indian Bioresource Information Network (IBIN) serves as a portal which networks the otherwise independent databases into a unified delivery system (http://ibin.gov.in/index.php?option=com_ibin&task= about). IBIN portal provides a single window for accessing myriad spatial/non-spatial data on bioresources of the country. This setup allows the data to be made available to a range of end users at a single end-point thereby ensuring that the data is always available in a consistent format thereby making it simple to consume.

To promote the reuse of IBIN species occurrence dataset among organizations in an interoperable format including support for integration across various platforms & programming languages, it is been exposed as web service using OGC Web Processing Service (WPS) standard. The OGC WPS provides a standardized interface for performing either simple or complex geoprocessing operation/computation online via web service from the remote host (Borges, 2015; Schut, 2007). As a result, reusability of the data in an interoperable manner is achieved, which is also platform-independent and can be consumed by multiple programming languages. This also provides the power to chain simple processes to allow for the execution of various complex processes in a variety of different contexts.

‘Species Occurrence Search’ WPS service takes inputs as either the species name or bounding box or shapefile defining the area of interest and returns queryable OGC complaint Web Map Service (WMS) as output with specie(s) occurrences represented in grid (5km x 5km) format, with each grid possessing attributes like specie(s) name, family, state, medicinal detail etc. In the following section, the reader will find the overall setup of this WPS architecture, its important features, the design and implementation of the ‘Species Occurrence Search WPS’ and of the JavaScript based web client and QGIS plugin which consume this WPS and use the WMS output to display results.

* Corresponding author

2. WEB PROCESSING SERVICE

A Web Processing Service (WPS) is a standardized interface defined by the Open Geospatial Consortium (OGC). It is a web service which makes it possible to execute computing processes and retrieve metadata which describe their purpose and functionality. The capabilities of a WPS can be retrieved using a GetCapabilities request, details of a specific service can be obtained using a DescribeProcess request while the processes can be executed using an Execute request (Borges, 2015; Schut, 2007). Since the release of version 2.0.0, job control and monitoring operations like GetStatus, GetResult and Dismiss have also been added which are particularly useful during an asynchronous execution.

2.1 Overall Architecture

The WPS standard forms the heart of this project. The open source ZOO-project was used to develop the ‘Species Occurrence Search’ WPS service.

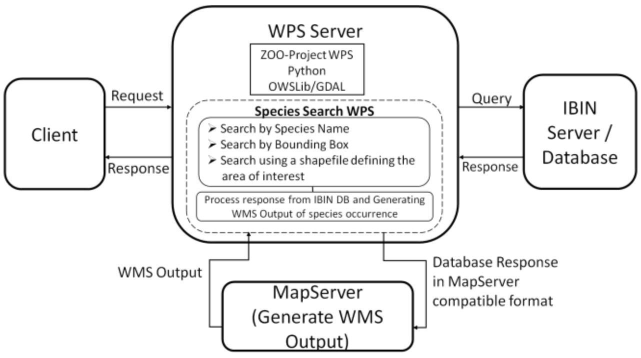
WPS Server
ZOO-Project WPS
Python
OWSLib/GDAL
Species Search WPS
➢ Search by Species Name
➢ Search by Bounding Box
➢ Search using a shapefile defining the area of interest
Process response from IBIN DB and Generating WMS Output of species occurrence
Client
Request
Response
Query
Response
IBIN Server / Database
WMS Output
Database Response in MapServer compatible format
MapServer (Generate WMS Output)

Figure 1. The overall architecture of the Species Search WPS

Figure 1 shows the overall architecture of the Species Search WPS. It allows searching for species occurrence using three ways either using species name or bounding box input or using a shapefile denoting the area of interest. WPS service is not just one service, this is actually a simplified representation for three services, one of which will be chained with WPS based on the inputs. WPS service processes the inputs using which it queries the IBIN database. Once it receives the response form the database, it converts the received response into a format that is accepted by MapServer (https://mapserver.org/index.html).

ZOO-project enables to write the WPS processes in languages like Python, PHP, Java, C# and JavaScript. Here the species search WPS service is written in Python using various Python geospatial libraries like GDAL, OWSLib etc. ZOO-project also provides a capability to integrate MapServer support (http://www.zoo-project.org/). This happens in such a way that once a WPS using MapServer support is terminated, its outputs are passed to MapServer. Once MapServer gives the WMS output, that output is received by WPS species search service which makes some necessary changes to the WMS so as to enable handling GetFeatureInfo requests. This enables user to get the features associated with each grid (species search result) of the output fetched from IBIN species occurrence database. Finally, the WMS output is returned to the client which can then be used by the client to visualize the output.

2.2 Important Features

2.2.1 Interoperability: A WPS allows the processes and code to be delivered to organizations irrespective of underlying program (Wehrmann et al., 2011). This ensures that the functionality can be used by organizations in a platform-independent manner while also giving the managing body to make necessary changes and updates to the code without breaking the functionality for any of the organizations.

2.2.2 Reusability: Services exposed as a WPS can be reused by organizations in multiple applications (Wehrmann et al., 2011). This means that the same functionality can be incorporated into multiple applications without having to explicitly design that functionality for each application separately, simply by importing the WPS into the application.

2.2.3 Service Chaining: This is a workflow of services where for each pair of services, the second service can occur only after the first one is terminated (Meng et al., 2009). This allows the creation of repeatable workflows and chunking of complex tasks into simpler blocks, each handled by a different service. Existing geospatial services like WMS or another WPS can also be incorporated into such a service chain (Meng et al., 2009).

2.2.4 Asynchronous execution: Processing of geospatial data often takes a long time. This can often exceed the maximum connection timeout range of Hyper Text Transfer Protocol (HTTP) servers, which is what WPS relies on (Čepický, 2008). Therefore, it is desirable to have an asynchronous execution of the same, as it decouples the request from the response and consequently avoids wasting and draining client resources till the processing goes on at the server end. (Westerholt and Resch, 2015).

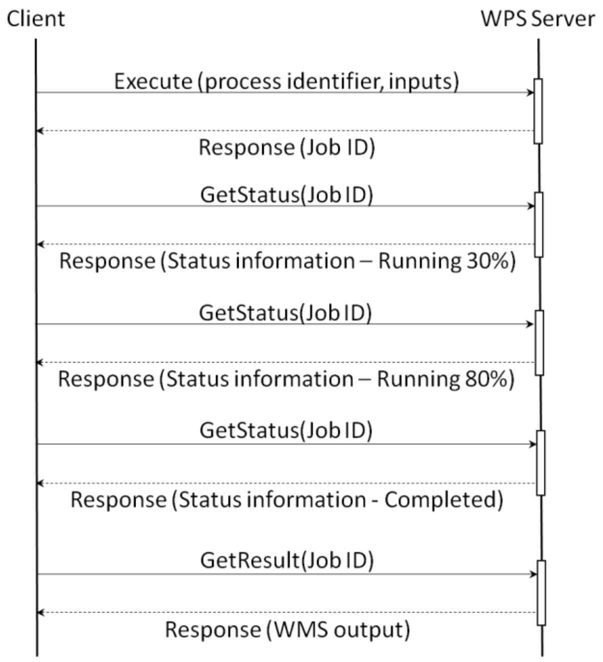
Client
WPS Server
Execute (process identifier, inputs)
Response (Job ID)
GetStatus(Job ID)
Response (Status information – Running 30%)
GetStatus(Job ID)
Response (Status information – Running 80%)
GetStatus(Job ID)
Response (Status information - Completed)
GetResult(Job ID)
Response (WMS output)

Figure 2. Asynchronous execution sequence diagram of the WPS process

Process management operations were added in WPS 2.0 (Borges, 2015). This allows the client to make an asynchronous Execute request, monitor the status of execution via a GetStatus request and once the execution is complete, make a GetResult request to fetch the result of execution from the WPS server. Figure 2 shows an asynchronous execution sequence as applicable for the Species Search WPS. The client would make an Execute request, passing the required input as either the name of a species or a bounding box or a shapefile defining the area of interest, and would be notified of a JobID for the process which has thus been initiated. The client would then constantly ping the server with GetStatus requests passing the JobID and would be notified of the status of execution as well as the percentage of completion of execution when the process is running. Finally, once the client is notified that the execution is completed, the client would retrieve the results by making a GetResult request passing the JobID, for which the WPS server would return the WMS output containing the results showing the location of species occurrence with attributes as per the input data provided.

2.3 Choice of WPS Framework

ZOO-project was selected as the framework for the Species Search WPS. A major driving factor was the support of multiple languages in Zoo, which is not provided by other frameworks like PyWPS (http://pywps.org/) which supports Python, 52° North (http://52north.org/communities/geoprocessing/wps/) which supports only Java, or GeoServer WPS (http://docs.geoserver.org/stable/en/user/services/wps/index.html) which again only supports Java. This gives flexibility to the maintaining organization to develop and publish other services in different languages as preferred by the developers. The performance of ZOO-Project is acceptable considering the tested response times, failure rates and throughput with concurrent requests. Further, among PyWPS and ZOO, frameworks which support Python, the performance of ZOO is reported to be better in all three metrics and is known to have a better support community (Poorazizi and Hunter, 2015).

3. DESIGN AND IMPLEMENTATION

3.1 Adding Capabilities for Handling Different Inputs

The service has been designed to take either the name of the species, a bounding box or a shapefile describing the area of interest as an input. This means that the WPS should be able to handle all these three types of inputs and process them accordingly.

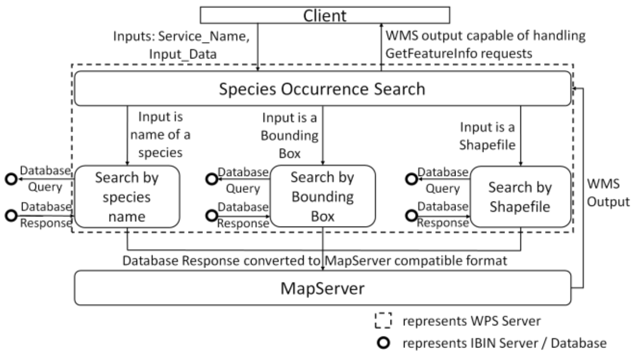
Client
Inputs: Service_Name, Input_Data
WMS output capable of handling GetFeatureInfo requests
Species Occurrence Search
Input is name of a species
Input is a Bounding Box
Input is a Shapefile
Database Query
Database Response
Search by species name
Search by Bounding Box
Search by Shapefile
WMS Output
Database Response converted to MapServer compatible format
MapServer
represents WPS Server
represents IBIN Server / Database

Figure 3. Species Occurrence Search WPS Service

To provide this functionality, the service is made to accept two inputs instead. The first input parameter, called Service_Name, asks the user for the choice of service to be executed. This defines the type of input that the user will be providing to the service. The second parameter, called Input_Data, is the parameter which accepts the name of the species or the coordinates of a bounding box or the URL of a shapefile as an input. Figure 3 shows the complete structure of WPS chaining occurring in the Species Search WPS. The Species Occurrence Search WPS validates the inputs, if the inputs are invalid, an error message is returned, otherwise one of the services – Search by species name, search by bounding box or search by shapefile, is chained with the existing WPS by passing to that service the required inputs from Input_Data. Here ‘Search by species name’, ‘Search by Bounding Box’ and ‘Search by Shapefile’ are the three services that form WPS as described in the overall architecture.

3.2 Querying the IBIN Database according to the Input

This part of the design uses selective chaining of processes. Based on the type of inputs passed, a separate process is executed which processes the inputs as required, makes the appropriate request to IBIN database, and receives the response from the database. This response contains data of all the locations where some species are found if the input was the name of a species, or all the species which were found within a bounding box or an area of interest and their locations. Further, available information about each species is also part of the response, like family, medicinal value etc. This response is in raw format which must be processed so that it can be returned to the client is a useful format.

3.3 Generating WMS Output

For a client, it would be useful if the data was returned in a format that represented all the data graphically instead of raw data which would require processing by the client for finding useful details from the output. This is where generating a WMS output comes in. A WMS output would return to the client a raster layer which can be overlaid onto a map. The Species Search WPS returns the location of species as 5km x 5km grids. The data associated with each grid can be accessed by passing the coordinates of some point in the grid as parameters in a GetFeatureInfo request to the WMS server. This removes all spatial processing load from the client, except displaying the WMS layer, and gives the data in a graphical format.

To make the WMS output, ZOO-project provides support for integrating MapServer with the WPS. This integration makes it possible to pass some data to MapServer for the generation of a WMS output. This output is not directly capable of handling GetFeatureInfo requests. To add this capability, the WPS was configured to make the necessary changes before publishing the output to the client. This ensures that the client can always fetch all the associated data at any point by making the corresponding GetFeatureInfo request to the WMS server, which the client can identify using the URL of the WMS output received by the client.

3.4 Developing Clients for Consuming the WPS

While the WPS provides the server-side functionalities which could be incorporated into multiple applications to suit the needs of organizations, it was imperative that some clients that can consume the WPS were developed. This was necessary so that end users who wish to gather the results could use the available clients, instead of manually making the request to the WPS and handling the WMS outputs. Consequently, a web client and a custom QGIS plugin were developed.

3.4.1 Web Client

Zoo-project provides boilerplate JavaScript code which is capable of handling WPS operations, both synchronous and asynchronous. The web client is built using this boilerplate code as a base (figure 4). It has been developed to make all requests asynchronously, and the user is notified of the progress via a progress bar which is updated with the response of each GetStatus request that is made. Leaflet (https://leafletjs.com/), an open-source JavaScript library for interactive maps has been used to render maps and the WMS output (showing the location of species occurrence in grid format). Further client has functionality to make GetFeatureInfo requests whenever someone clicked on the WMS output layer. The results (showing attribute of species found) are then shown as a popup as shown in figure 5.

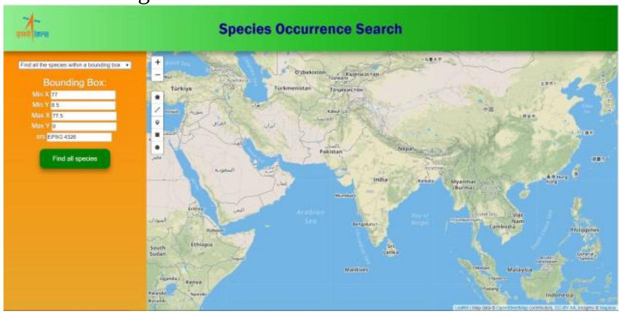
Species Occurrence Search
Find all the species within a bounding box
Bounding Box:
Min X
Min Y
Max X
Max Y
srs
EPSG:4326
Find all species

Figure 4. WPS Web Client- Species Occurrence Search by Bounding Box

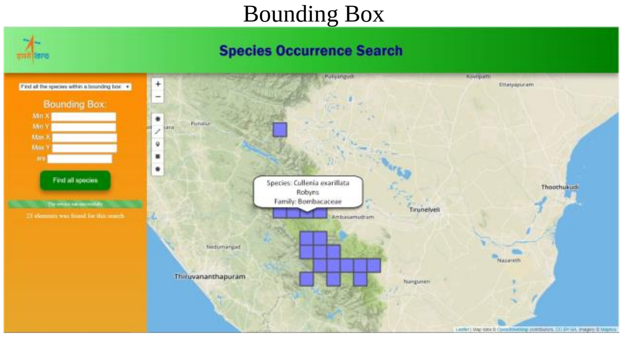

Figure 5. WPS Web Client- Output of Species Occurrence Search using user defined Bounding Box

3.4.2 Custom QGIS Plugin

The QGIS plugin for QGIS 2.18 was also developed as software like QGIS are commonly used for interpreting spatial data (figure 6).

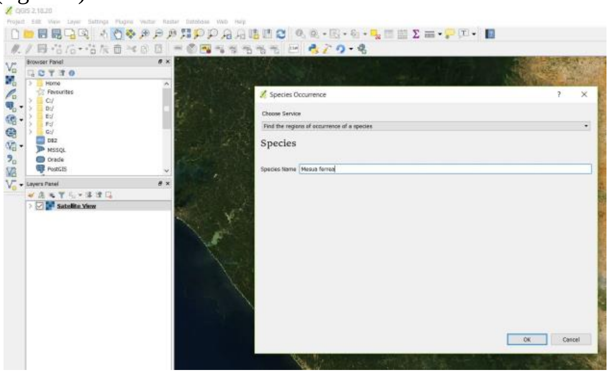

Figure 6. QGIS Plugin for IBIN Species Occurrence WPS Service

The plugin uses OWSLib (https://github.com/geopython/OWSLib) at its core, to make it capable of handling asynchronous execution. Once a request is made, the user is notified of a running process by a progress bar being displayed as a message. Upon successful completion of WPS process, the WMS layer denoting the output is added to the workspace as a layer, whose name the user is prompted to supply before the layer is added. The details of species associated with each grid can be seen by using the ‘Identify Features’ tool (figure 7).

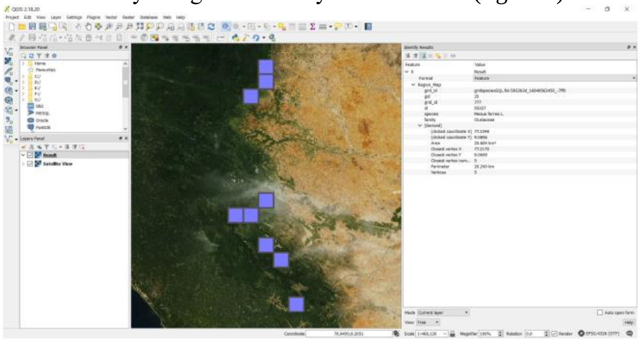

Figure 7. Executing IBIN Species Search WPS Service using Custom QGIS Plugin

4. CONCLUSION

The WPS for species occurrence search provides a way to access information of occurrence data of all the species of the country through one unified place. This provides data in a consistent manner to all users, thus eliminating issues of different format of outputs from different databases. Further, the data is supplied in a reusable and interoperable way. This ensures that the service can cater to the needs of the maximum number of users by surpassing any restrictions that may be imposed by platform, therefore, supporting extensively in further studies relating to species occurrence.

REFERENCES

Ball-Damerow, J., Brenskelle, L., Barve, N., Soltis, P., LaFrance, R., Ariño, A., Guralnick, R., 2017. Use of Online Species Occurrence Databases in Published Research since 2010, in: *Proceedings of TDWG*. p. e20518. https://doi.org/10.3897/tdwgproceedings.1.20518

Borges, a V.K., 2015. OGC WPS 2.0.2 Interface Standard: Corrigendum 2. *Open Geospatial Consortium*. https://doi.org/http://www.opengeospatial.org/

Čepický, J., 2008. Ogc Web Processing Service and It’S Usage. *GIS Ostrava* 2008 27, 1–12. https://doi.org/10.1007/springerreference_62558

Chapman, A.D. 2005., 2005. Uses of primary species-occurrence data, version 1.0 *Report for the Global Biodiversity Information Facility*

Dubois, G., Schulz, M., Skøien, J., Bastin, L., Peedell, S., 2013. eHabitat, a multi-purpose Web Processing Service for ecological modeling. *Environmental Modelling and Software* https://doi.org/10.1016/j.envsoft.2012.11.005

Elith, J., H. Graham, C., P. Anderson, R., Dudík, M., Ferrier, S., Guisan, A., J. Hijmans, R., Huettmann, F., R. Leathwick, J., Lehmann, A., Li, J., G. Lohmann, L., A. Loiselle, B., Manion, G., Moritz, C., Nakamura, M., Nakazawa, Y., McC. M. Overton, J., Townsend Peterson, A., J. Phillips, S., Richardson, K., Scachetti-Pereira, R., E. Schapire, R., Soberón, J., Williams, S., S. Wisz, M., E. Zimmermann, N., 2006. Novel methods improve prediction of species' distributions from occurrence data. *Ecography* (Cop.). https://doi.org/10.1111/j.2006.0906-7590.04596.x

Meng, X., Bian, F., Xie, Y., 2009. Geospatial Services Chaining with Web Processing Service, in: *International Symposium on Intelligent Information Systems and Applications (IISA'09)*.

Poorazizi, M.E., Hunter, A.J.S., 2015. Evaluation of Web Processing Service Frameworks. *OSGeo J*. 14, 29–42.

Schut, P., 2007. OpenGIS ® Web Processing Service. *Open Geospatial Consortium*. https://doi.org/citeulike-article-id:8653309

Stewart, F.E.C., Fisher, J.T., Burton, A.C., Volpe, J.P., 2018. Species occurrence data reflect the magnitude of animal movements better than the proximity of animal space use: *Ecosphere*. https://doi.org/10.1002/ecs2.2112

Wehrmann, T., Gebhardt, S., Klinger, V., Künzer, C., 2011. Data processing using Web Processing Service orchestration within a Spatial Data Infrastructure, in: *Proceedings of the 34th International Symposium on Remote Sensing of Environment*.

Westerholt, R., Resch, B., 2015. Asynchronous Geospatial Processing: An Event-Driven Push-Based Architecture for the OGC Web Processing Service. *Transactions in GIS* https://doi.org/10.1111/tgis.12104