Extension Designer Best Practices

This section includes best practices, advanced usage tips, and other information for help you implement Enterprise Pack applications.

Using Categories

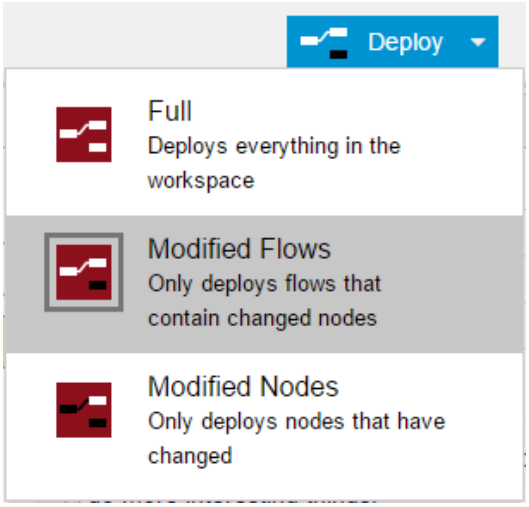
You should organize services into categories that describe the service. Categories are a tool for organizing services so that they are easier to find and maintain, but it is ultimately your discretion how the services are organized. By default Extension Designer includes three built-in categories:

- Process Intelligence Engine
- DTP Workflows
- Compliance Practices

These categories cannot be removed and are used internally, but services can be added and removed as desired.

Deploying

Because multiple users can view and edit a service at the same time it is recommended to change the deploy settings to **Modified Flows** or **Modified Nodes** rather that **Full Deploy** to avoid deploying over another users work.



Avoid Forking Flows

Forking flows can be detrimental to performance if the **msg** in the flow is large. A forking flow is any flow that has multiple wires that are attached to an output point, or a flow that has a node that has multiple output points that are invoked by a single message. When a **msg** is passed into a fork it is unclear how it will be mutated, because of this if the **msg** is cloned N-1 times, where N is the number of wires or output points the **msg** object is expected to pass through. The cloning of the **msg** simplifies slice creation for the end user by eliminating race conditions that can be introduced by a forking flow, so it is recomended to avoid forking as much as possible when working with slices that use a large amount of data.

Debug Your Flows

The **debug** node is extremely useful for debugging flows, but the node can affect performance. Because objects that are debugged have to be serialized and transmitted to the web browser, debugging large objects should be avoided whenever possible. You should instead focus on smaller pieces of the **msg** object as you debug.

For instance, if you want to verify that an array is located on a specific property of the msg object, you could debug the length of the array (e.g. msg.array.length) instead of debugging the entire array that could be thousands of elements long. Adding debug nodes generally creates a fork in the flow (unless it is the last node in the flow), so we recommend removing any debug nodes from the flow once they are no longer needed. Disabling the debug node, however, will not prevent the **msg** object from being forked and passed to the debug node. Disabling the node will only prevent the debug node from printing, resulting in unnecessary overhead.

Dividing Services: How Many Services Do I Need?

Each service will consume a maximum of 2GB RAM and 1 CPU. In many cases, you will run Extension Designer with many flows deployed to one service. We recommend organizing flows into the built-in categories, however, starting with one service per category. If you experience degraded performance, you can add another service to each category. If you notice high CPU usage associated with node processes, then you should spread flows into multiple services.

Each service has dedicated port (see Getting Started with Enterprise Pack). You should be able to detect from your OS's Process Manager which service has heavy loads (defined as nearly hitting 1.5GB RAM and constant 50% CPU). If this occurs, you should spread the flows out into multiple services to disperse the load.

Use HTTP(S)-compliant Parameters When Calling Endpoints

Some artifacts require you to call the endpoint in the browser or by using a cURL command. Modern browsers attempt to resolve non-compliant URL characters, but are not as reliable as manually ensuring that the characters are properly encoded.When adding your parameters, be sure to properly encode the string parameter values if they contain spaces, +, /, or any other characters that are not allowed in the HTTP(S) protocol.

The following parameter, for example, is not allowed and will prevent the artifact from functioning correctly: `buildId=c++test`. The encoded parameter would be `buildId=C%2D%2Dtest`.

You can use an encoding tool (e.g., http://www.url-encode-decode.com) to help you properly encode parameters to be compliant with the standard.

Using the Correct Endpoint Path

Service Designer makes two endpoint paths available in the Available Endpoints page

Additional Resources

You can check the Parasoft forum for answers from the community. You can also contact Parasoft Support and Parasoft Professional Services.