No XML Toolkit jobs will run if one program is stuck in MSGW status

Issue

When a command or program is called from the XML Toolkit, and the command or program halts on an inquiry error message (the job goes into MSGW status), the job running that command or program goes to a MSGW status, and cannot continue until the message is answered. Sometimes when one XML Toolkit job goes into the MSGW status, no other XML Toolkit calls will execute until the waiting message is responded to. This can happen when using a stateful connection.

Environment

Zend Server for IBM i XML Toolkit, any version running on any supported version of IBM i.

Resolution

Determine if you are using a stateful connection. The way to tell is if the InternalKey option is set and the stateless option is not set:

```
$ToolkitServiceObj->setToolkitServiceParams(array('InternalKey'=>'/tmp/$user'));
```

In the above example, the InternalKey option is set to '/tmp/$user', making this a stateful connection. A stateful connection starts a job in the ZENDPHP7 subsystem that services all requests for that InternalKey. So, if you use the same InternalKey for all your connections, and that job is tied up, no other XML Toolkit connection using that same InternalKey can execute.

If you are running stateful (the InternalKey option is set, or there is not a stateless option set), try running stateless:

```
$ToolkitServiceObj->setToolkitServiceParams(array('stateless'=>true));
```

According to the documentation, stateless is false by default, so even if you do not specify an InternalKey option, you are probably still running stateful using the default InternalKey. If none of your scripts are stateless, and none of them set an InternalKey, then all of your scripts are using the default stateful key, and all of your XML Toolkit requests are being processed by one job.

XML Toolkit options are described here:

Setting XML Toolkit Options

A final thought

Never run a program via the Toolkit that will halt on a message wait. There is really no excuse for it. ;-) Either handle the error in your RPG or COBOL program using basic error handling techniques, or call the program via a CLP that has a MONMSG in it to get you past the error.

Here are some error handling operations available in RPG:

Error Handling Operations

If you can't get at the RPG code for some reason (legacy or third party program, for example), call your RPG program via a CLP, and use MONMSG right after the CALL:

MONMSG command reference