27.0 Biz Service Authorization APIs

WAS THIS PAGE HELPFUL? Leave Feedback

27.0 Biz Service Authorization APIs

27.1 Get Service Authorization

Get authorized hosts, services, and service hosts based on authorized host and service groups. Authorized host and service groups are generally set by user role memberships managed externally. This service translates specified groups into a collection of hosts that have authorized access to all services and a map of host collections keyed by service that authorizes access to individual services on hosts. General API conventions are followed wrapping POST data and results for XML and JSON content types.

27.1.1 Method: POST Get Authorized Services

POST /api/biz/getauthorizedservices

27.1.2 POST Data Example

Here is an XML example post data for accessing host group authorized service:

```
<bizAuthorization>
    <hostGroupNames>
        <hostGroupName>Linux Servers</hostGroupName>
    </hostGroupNames>
</bizAuthorization>
```

Here is an XML example of post data for accessing service group authorized services:

```
<bizAuthorization>
    <serviceGroupNames>
        <serviceGroupName>local</serviceGroupName>
    </serviceGroupNames>
</bizAuthorization>
```

Here is a JSON example post data for accessing both host and service group authorized services:

```
{
    "hostGroupNames" : ["Linux Servers"],
    "serviceGroupNames" : ["local"]
}
```

27.1.3 HTTP Headers

Header	Valid Values	Required
Content-Type	application/xml or application/json	False
Accept	application/xml or application/json	False
GWOS-API-TOKEN	a valid token returned from login	True
GWOS-APP-NAME	your application name	True

27.1.4 HTTP Status Codes

Code	Description
200	Authorized hosts and services returned
401	Authentication/authorization error occurred
404	No authorization returned, (unlimited access can be assumed)
500	An internal server error occurred while querying host blacklists

27.1.5 Example Results

Here is an XML example of authorized hosts and services results. Note that service host maps are verbose when marshalled into XML. General API conventions are followed wrapping results with XML tags.

```
<bizAuthorizedServices>
    <hostNames>
        <hostName>localhost</hostName>
    </hostNames>
    <serviceHostNames>
        <serviceHostMap>
            <serviceHostMapEntry serviceName="local_cpu_httpd">
                <hostNames>
                    <hostName>localhost</hostName>
                </hostNames>
            </serviceHostMapEntry>
            <serviceHostMapEntry serviceName="local_cpu_java">
                <hostNames>
                    <hostName>localhost</hostName>
                </hostNames>
            </serviceHostMapEntry>
            <serviceHostMapEntry serviceName="local_cpu_syslog-ng">
                <hostNames>
                    <hostName>localhost</hostName>
                </hostNames>
            </serviceHostMapEntry>
            <serviceHostMapEntry serviceName="local_swap">
                <hostNames>
                    <hostName>localhost</hostName>
                </hostNames>
            </serviceHostMapEntry>
            <serviceHostMapEntry serviceName="local_process_nagios">
                <hostNames>
                    <hostName>localhost</hostName>
                </hostNames>
            </serviceHostMapEntry>
            <serviceHostMapEntry serviceName="tcp_http">
                <hostNames>
                    <hostName>localhost</hostName>
                </hostNames>
            </serviceHostMapEntry>
        </serviceHostMap>
    </serviceHostNames>
</bizAuthorizedServices>
```

Here is a JSON example of authorized hosts and services results. General API conventions are followed wrapping results with JSON objects.

```
{
  "hostNames" : [ "localhost" ],
  "serviceHostNames" : {
    "local_cpu_httpd" : [ "localhost" ],
    "local_cpu_java" : [ "localhost" ],
    "local_cpu_syslog-ng" : [ "localhost" ],
    "local_swap" : [ "localhost" ],
    "local_process_nagios" : [ "localhost" ],
    "tcp_http" : [ "localhost" ]
  }
}
```