4.1.x Cartridge Group Resource Definition

A cartridge group defines the relationship among a set of cartridge groups and a set of cartridges. The relationship among the children of a group can be the startup order, termination behavior and any scalable dependencies. Writing a cartridge group definition provides the ability to re-use the same frequently used cartridges as needed in different composite applications. The cartridges that correspond to a cartridge group have to be added to Stratos before the cartridge group is added. A sample cartridge group definition, together with information on all the properties that can be added in a cartridge group definition JSON are as follows:

- Sample cartridge group definition JSON
- Property definitions

Sample cartridge group definition JSON

The following are sample configurations that can be used in a JSON to define a cartridge group:

```
{
    "name": "esb-php-group",
    "cartridges": [
        "esb",
        "php"
    ],
    "dependencies": {
        "startupOrders": [
            {
                "aliases": [
                    "cartridge.my-esb",
                    "cartridge.my-php"
                ]
            }
        ],
        "terminationBehaviour": "terminate-none"
    }
}
```

Property definitions

All the properties that correspond to the cartridge group resource are explained as follows:

- name
- groups
  - name
  - groups
- cartridges
- dependencies
  - startupOrders
  - scalingDependents
  - terminationBehaviour
- cartridges
- dependencies
  - startupOrders
  - scalingDependents
  - terminationBehaviour
### Main property definitions

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
<th>JSON</th>
<th>UI</th>
<th>Writable</th>
<th>Readable</th>
<th>Updatable</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the group.</td>
<td>Yes</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>groups</td>
<td>Defines all other nested groups within the group.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cartridges</td>
<td>If the group refers to any cartridges, then such details needs to be mentioned under this parameter. When defining cartridges, you need to use the name of the cartridge.</td>
<td>Yes</td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
dependencies | Dependencies | Defines the dependency that exists between members (cartridge or a group). For more information on the sub-properties, see dependencies. | Yes | N/A

Sub-property definitions

dependencies

<table>
<thead>
<tr>
<th>Property</th>
<th>JSON</th>
<th>UI</th>
<th>Writable</th>
<th>Readable</th>
<th>Updatable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startupOrders</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>The order in which the (sub groups and cartridge) should be started up. If it is a group, it should use group.&lt;GROUP_ALIAS&gt; cartridge, it should be &quot;cartridge.&lt;CARTRIDGE_ALIAS&gt;&quot; when defining startupOrders. Multiple startupOrders can be defined as String array startupOrder should not create a cyclic dependency. By defining startupOrders, parallel dependencies can be identified to start them in parallel.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- If the `startupOrder` is as follows, group3 should come before mcat and php. However, group1 can come up in any order.

  ```json
  "startupOrders": [
    {
      "aliases": [
        "group.group3",
        "cartridge.tomcat"
      ]
    },
    {
      "aliases": [
        "group.group1",
        "cartridge.php"
      ]
    }
  ],
  ```

- If the `startupOrder` is as follows, group1 will come up first, then tomcat and then php.

  ```json
  "startupOrders": [
    {
      "aliases": [
        "group.group1",
        "cartridge.tomcat",
        "cartridge.php"
      ]
    }
  ],
  ```
If the startupOrder has two independent rules defined as follows, then once group1 is brought up, group1 and cartridge can be brought up in parallel.

```json
"startupOrders": [
    {
        "aliases": [
            "group.group1",
            "cartridge.tomcat"
        ]
    },
    {
        "aliases": [
            "group.group1",
            "cartridge.php"
        ]
    }
]
```

If a startupOrder is not defined then the groups or cartridges will be started up parallel to each other.
<table>
<thead>
<tr>
<th>terminationBehaviour</th>
<th>Termination Behaviour</th>
</tr>
</thead>
</table>

This determines how the instances need to be terminated. The following is an example startupOrder used to explain the terminationBehaviour:

```
"startupOrders": [
  {
    "aliases": [
      "group.group1",
      "cartridge.tomcat"
    ]
  },
  {
    "aliases": [
      "group.group1",
      "cartridge.php"
    ]
  }
],
```

The available termination behaviours are as follows:

- **terminate-none**
  None of the children in a group will be terminated. For example, if something happens to `group1`, it will not have an impact on `tomcat` or `php`.

- **terminate-all**
  All the elements in the dependency tree will be terminated. For example, if something happens to `php`, then all the children of the group `php` will be terminated regardless of the other dependency information. Note that cartridges and sub-groups are considered as the children of a group.

- **terminate-dependents**
  Only the dependents will be terminated. For example:
  - If something happens to `group1`, then `tomcat` and `php` will get terminated. However, if any other cartridge or group is found in the parent group that belongs to `group1`, then that cartridge or group will remain as it is in Stratos.
  - If something happens to `php`, nothing will get terminated as `php` does not have any
| scalingDependents | Scaling Dependents | Defines if there are any dependencies with regard to scaling. When dependencies are defined among members (cartridge or a group), and scaling (scaling up or down) is taking place for any of the members, all other dependent members will also scale in order to maintain the defined ratio. For more information, see Scaling. |