**Technical Info for**

**Server Mediator Utility Classes**

TOC

[مقدمه 2](#_Toc432065114)

[Internal type 2](#_Toc432065115)

[Ping-Pong 2](#_Toc432065116)

[معرفی سرویس 2](#_Toc432065117)

[Communication Types 3](#_Toc432065118)

[Call 3](#_Toc432065119)

[Callback 4](#_Toc432065120)

[Fire 4](#_Toc432065121)

[Predefined Nodes 4](#_Toc432065122)

[Node های داخلی 4](#_Toc432065123)

[Node های ثابت سرویس ها 5](#_Toc432065124)

# مقدمه

در اینجا در مورد اطلاعاتی که مابین سرور و سرویس ها رد و بدل میشود توضیح میدهیم.

همانطور که گفته شده تمامی داده در قالب JSON هستند و هر پیغام در قالب یک struct بدون نام JSON . همر پیغام یک فیلد type دارد که نوع ساختار آن پیغام را مشخص میکند. چهار نوع پیغام وجود دارد: internal ، call ، callback ، fire . نوع internal بسیار محدود و برای انجام دستورات سطح پایین مانند معرفی سرویس مورد استفاده قرار میگیرد و سه نوع دیگر برای کار با Command ها و Event ها .

همچنین علاوه بر فیلد type ، فیلد دیگری هم با نام node وجود دارد که کاربرد آن مشخص کردن هدف است. جزییات بیشتر را در هر بخش خواهید دید.

نکته: نام node ها case sensitive نیست.

# Internal type

این نوع در موارد زیر کاربرد دارد که در هر بخش جزییات آن مشخص میشود:

## Ping-Pong

همانند ping عمل میکند. دستور ping یک درخواست است که هر کدام از طرفین آن را دریافت کردند باید با pong به آن جواب دهند. جزییات فیلد ها:

|  |  |  |
| --- | --- | --- |
| **Description** | **Value** | **Field** |
|  | “Internal” | type |
|  | “ping” / “pong” | node |
| حداکثر 8 کاراکتر | یک مقدار تصادفی | id |

\*مثال:

|  |  |
| --- | --- |
| {“type” : “internal” , “node” : “ping” , “id” : “125684”} | Request |
| {“type” : “internal” , “node” : “pong” , “id” : “125684”} | Response |

## معرفی سرویس

قبل از هر نوع ارتباطی با سرویس و برای شناسایی سرویس مورد استفاده قرار میگیرد. ابتدا درخواست توسط سرور ارسال میشود:

|  |  |  |
| --- | --- | --- |
| **Description** | **Value** | **Field** |
|  | “Internal” | type |
|  | “hello” | node |

و سرویس باید به این شکل به آن جواب دهد:

|  |  |  |
| --- | --- | --- |
| **Description** | **Value** | **Field** |
|  | “Internal” | type |
|  | “hello” | node |
|  | اسم سرویس | name |
|  | ورژن سرویس | version |
| اگر سرویس قبلا نصب شده باشد باید این کد را که هنگام نصب دریافت کرده ارائه دهد. در غیر این صورت به عنوان یک سرویس نصب نشده به حساب میآید. | کد شناسایی سرویس | id |

# Communication Types

که برای اجرای Command ها و Event ها مورد استفاده قرار میگیرند. تمامی دستورات مدیریتی سرور هم توسط این نوع ها و node های داخلی سرور انجام میشود. Node های داخلی را در بخش های بعدی معرفی میکنیم.

جزییات این سه نوع:

## Call

دستوری است که یک command را فراخوانی میکند. از سرور به سرویس و بلعکس.

|  |  |  |
| --- | --- | --- |
| **Description** | **Value** | **Field** |
|  | “call” | type |
|  | command name | node |
| یک مقدار تصادفی است که برای مدیریت callback ها مورد استفاده قرار میگیرد. حداکثر 8 کاراکتر. | Random value | id |
| تنها در حالت از سرور به سرویس این فیلد وجود دارد که سرویس درخواست دهنده را مشخص میکند | درخواست دهنده | from |
| کد session که برای شناسایی کاربر درخواست دهنده این دستور و سطح دسترسی مربوطه مورد استفاده قرار میگیرد. | Random value | session |
|  | Command input | data |

کاربرد فیلد id برای برگرداندن مقدار بازگشتی command اجرا شده توسط callback ها است. به این شکل که هنگامی که یک command مقدار بازگشتی داشته باشد، یک پیغام از نوع callback به سرویس با آن مقدار بازگشتی و id که برای فراخوانی آن command استفاده شده، فرستاده میشود. که با استفاده از id میتوان تشخیص داد که این callback مربوط به کدام call است.

سرویس مقصد **باید** مقدار بازگشتی خود را با همان id مشخص کند. البته لازم به ذکر است که مقدار id از سرویس مبدا به سرور و از سرور به سرویس مبدا یکسان است، همچنین مقدار id از سرور به سرویس مقصد و از سرویس مقصد به سرور نیز یکسان است، ولی مقدار id مابین سرویس مبدا و مقصد یکسان نیست.

## Callback

مقدار بازگشتی یک command است.

|  |  |  |
| --- | --- | --- |
| **Description** | **Value** | **Field** |
|  | “callback” | type |
| همان نام command صدا زده شده است | command name | node |
| حداکثر 8 کاراکتر. | Random value | id |
| تنها در حالت از سرور به سرویس این فیلد وجود دارد که سرویس درخواست دهنده را مشخص میکند | درخواست دهنده | from |
|  | Command output | data |

## Fire

برای فراخوانی یک رویداد ( از سرویس به سرور ) و همچنین برای فراخوانی hook های متصل به آن رویداد ( از سرور به سرویس ها ) به کار میرود.

|  |  |  |
| --- | --- | --- |
| **Description** | **Value** | **Field** |
|  | “fire” | type |
|  | Event name | node |
| تنها در حالت از سرور به سرویس این فیلد وجود دارد که سرویس درخواست دهنده را مشخص میکند | درخواست دهنده | from |
|  | Event data | data |

# Predefined Nodes

برای انجام عملیات داخلی سیستم و مدیریت سرویس ها تعدادی node از قبل تعریف شده اند ( سمت سرور و سمت سرویس ) که همیشه وجود دارند. تعریف و استفاده از نود های سمت سرویس بر عهده توسعه دهنده آن سرویس است.

## Node های داخلی

نکته : نود های داخلی سرور همه پیشوند “\_Core.” دارند.

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Data field’s fields** | **type** | **Name** |
|  | settingName, newValue | Event | onSettingUpdate |
|  | names[] | Command | registerCommand |
|  | names[] | Command | removeCommand |
|  | names[] | Command | registerEvent |
|  | names[] | Command | removeEvent |
|  | names[] | Command | registerHook |
|  | names[] | Command | removeHook |
|  | node,[id],description | Command | error |
| Level=[note ,debug, warning, critical] | level, node,[id],description | Command | warning |
|  | name ,defValue ,public ,readonly | Command | registerSetting |
|  | name | Command | removeSetting |
|  | name, value | Command | setSetting |
| Return: name, value | name | Command | getSetting |
|  | name ,defValue ,public ,readonly | Command | updateSetting |
| Return: name, value | name | Command | resetSetting |

## Node های ثابت سرویس ها

این نود ها باید همیشه سمت سرویس وجود داشته باشند و نیازی به register کردن آنها در زمان نصب نیست. همچنین این نود ها پیشوند ندارند.

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Data field’s fields** | **type** | **Name** |
|  | Id | Command | onInstall |
|  |  | Command | onUninstall |
|  |  | Command | onEnable |
|  |  | Command | onDisable |
| Return:  name ,  title ,  version ,  icon , //optional  commands:[{  name,  inputDatatype,  inputDatatypeVersion,  outputDatatype,  outputDatatypeVersion  },…],  events:[{  name,  datatype,  datatypeVersion,  },…],  hooks: [{  name,  datatype,  datatypeVersion,  },…],  calls : [{  name,  inputdatatypeversion,  outputdatatypeversion  },…] ,  requirements:[{  name,  version  },…],  datatypes: [ {  name,  version  } , ... ] |  | Command | getInstallInfo |
| Return:  Datatype,newversion,Value | value, datatype,  fromVersion,  toVersion | Command | changeDatatypeVersion |
| Optional | node,[id],description | Command | error |
| Optional,  Level=[note ,debug, warning, critical] | level, node,[id],description | Command | warning |