**­**

**Documentation**

**for the**

**Eiffel Backend Service**

Compilation

Execution

Flat View

Contract View

Class Descendants

Class Ancestors

Class Clients

Class Suppliers

Feature Callers

Command Line

Compilation and Runtime Timeouts

Status Codes Used

Removing old Projects

Apache Configuration

Assumptions

**Compilation**

[http://localhost:9090/compile?clean=true;path=your\_path;id](http://localhost:9090/compile?clean=true;path=)=your\_id;target=your\_target

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter Name | Optional | Default Value (if optional) | Type |
| clean | Yes | false | Boolean |
| path | Not for the first time a project is compiled |  | String |
| id | Only for the first time a project is compiled. After that you always have to supply a VALID id |  | String |
| target | If the project has multiple targets, then it is non-optional for the first time. If the project has only a single target, then this field is optional. |  | String |

The first time you use a project, you have to provide a non-empty ‘path’. The ‘id’ field may be blank or not present. The clean field may be true/ false or not present. The target field must be non-empty for a multiple-target project, otherwise it can be blank or not present in a single target project.

The next time, you have to provide a VALID id. You may omit ‘clean’, and the ‘path’ may be empty or not present. The target may be blank or not present, but if you want to change the target, you can specify it.

The first time a project is compiled, it will always undergo clean compilation.

The server will create a new project directory for you in ./projects/ and return the unique id in the “id” key of the response object.

If the ‘id’ is invalid, the server responds with a 410 Gone Status Code in the response message.

**Targets:** Whenever you provide a target, the server stores it and uses it as the default target for all other functions from the API, unless you specify a new target and compile using that new target.

You can only change your default target by calling /compile with the new target. No other calls change the default target.

If the target is invalid, the server responds with the list of available targets. You MUST compile with the right target first, and then use any other functions from the API. If the server has a wrong target stored and you try to use other functions from the API, it will result in an error.

**Note**: Whenever you provide a path in the arguments, it will always do a clean compilation of the project.

The response object JSON for compile looks something like this:

[{

"Compile\_Message":"Eiffel Compilation Manager\nVersion 14.05.9.5158 GPL Edition - win64\n\nDegree 6: Examining System\nDegree 5: Parsing Classes\nDegree 4: Analyzing Inheritance\nDegree 3: Checking Types\nDegree 2: Generating Byte Code\nFreezing System Changes\nDegree -1: Generating Code\nSystem Recompiled.",

"Output\_Message":"Eiffel C/C++ Compilation Tool - Version: 14.05\nCopyright Eiffel Software 1996-2012. All Rights Reserved",

"Error\_Message":"",

"Has\_Compilation\_Error":false,

"Warning\_Message":"Warning code: Unused\_local\_warning\n\nWarning: unreferenced local variable(s)\nWhat to do: Remove it if you don't plan to use it in the future.\n\nClass: APPLICATION\nFeature: make\nUnused local is: \n\tc: INTEGER\_32\n\n-------------------------------------------------------------------------------",

"Has\_Warning":true,

"Needs\_Target":false,

"Dump ":"Eiffel Compilation Manager\nVersion 14.05.9.5158 GPL Edition - win64\n\nDegree 6: Examining System\nDegree 5: Parsing Classes\nDegree 4: Analyzing Inheritance\nDegree 3: Checking Types\nDegree 2: Generating Byte Code\nFreezing System Changes\nDegree -1: Generating Code\n-------------------------------------------------------------------------------\n\nWarning code: Unused\_local\_warning\n\nWarning: unreferenced local variable(s)\nWhat to do: Remove it if you don't plan to use it in the future.\n\nClass: APPLICATION\nFeature: make\nUnused local is: \n\tc: INTEGER\_32\n\n-------------------------------------------------------------------------------\nSystem Recompiled.\nYou must now run \"finish\_freezing\" in:\n\tC:\\Users\\Manav\\Desktop\\eve\_server\\projects\\sample\_90799D1A-C197-4822-BEE9-C0D076CA8562\\EIFGENs\\sample\\W\_code\n",

"Errors":null,

"Warnings":[{

"Warning\_Code":"Unused\_local\_warning\n",

"Warning":"unreferenced local variable(s)\n",

"What\_to\_do":"Remove it if you don't plan to use it in the future.\n\n",

"Class":"APPLICATION\n",

"Feature":"make\n",

"After\_Feature":"Unused local is: \n\tc: INTEGER\_32\n\n",

"Dump":"Warning code: Unused\_local\_warning\n\nWarning: unreferenced local variable(s)\nWhat to do: Remove it if you don't plan to use it in the future.\n\nClass: APPLICATION\nFeature: make\nUnused local is: \n\tc: INTEGER\_32\n\n"

}],

"Targets":null,

"Compilation\_Succeeded":true

}]

The keys in the JSON response for compilation

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Key | Type | Description | Default Values |
| Compile\_Message | String | Gives the compile message | “” |
| Output\_Message | String | Gives the C compilation results and any other output messages if any | “” |
| Error\_Message | String | Gives the unparsed error message | “” |
| Has\_Compilation\_Error | Boolean | Indicates whether there is an error or not in the compilation | True/false |
| Warning\_Message | String | Gives the unparsed warning message | “” |
| Has\_Warning | Boolean | Indicates whether there is a warning message in the compilation. | True/false |
| Needs\_Target | Boolean | Indicates that the user either specified a wrong target or did not specify a target for a multi-target project.  This field MUST always be false before you use another function from the API, otherwise it will give an error | True/False |
| Dump | String | Gives the entire dump message with errors, warnings, and the compile results. | “” |
| Errors | Json Array | Gives the parsed JSON object for the error message with its keys | Null |
| Warnings | Json Array | Gives the parsed JSON object for the warning message with its keys | Null |
| Targets | Json Array | Gives the parsed JSON object for the list of available targets for the project | Null |
| Compilation\_Succeeded | Boolean | Indicates whether the compilation was successful or not.  The compilation fails under the following circumstances:   1. Compilation Error : You can check the Has\_Compilation\_Error field, which will be true. 2. Wrong/ Missing Target: You can check the Needs\_Target field, which will be true. 3. Timeout: You can check the status code for timeouts (504). | True/False |

The keys for the Warnings in the JSON response for compilation

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Key | Type | Description | Default Value(if not present in the warning message) |
| Warning\_Code | String | Gives the warning code | “” |
| Warning | String | Gives the warning message | “” |
| What\_to\_do | String | Gives the what to do string | “” |
| Class | String | Gives the name of the class where the warning occured | “” |
| Feature | String | Gives the name of the feature where the warning occured | “” |
| After\_Feature | String | Gives the part after the feature in the warning message | “” |
| Dump | String | Gives the dump of this warning message | Has to be present |

The keys for the Error in the JSON response for compilation

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Key | Type | Description | Default Value(if not present in the error message) |
| Error\_Code | String | Gives the error code | “” |
| Error | String | Gives the error message | “” |
| What\_to\_do | String | Gives the what to do string | “” |
| Class | String | Gives the name of the class where the error occured | “” |
| Feature | String | Gives the name of the feature where the error occured | “” |
| Line | Integer | Gives the line number where the error occured | -1 |
| Before\_Line | String | Gives the part before the line in the error message | “” |
| After\_Line | String | Gives the part after the line in the error message | “” |
| Dump | String | Gives the dump of this error message | Has to be present |

The keys for the Targets in the JSON response for compilation

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Key | Type | Description | Default Value(if not present in the error message) |
| Target | String | Gives the target | “” |

Target JSON\_array

"Targets":[{"Target":"sampleA"},{"Target":"sampleB"}]

An Example of an error message

[{

"Compile\_Message":"Eiffel Com",

"Output\_Message":"",

"Error\_Message":"Error code: VEEN\n\nError: unknown identifier.\nWhat to do: make sure that identifier, if needed, is final name of\n feature of class, or local entity or formal argument of routine.\n\nClass: APPLICATION\nFeature: extra\_feature\nIdentifier: asdas\nTarget type: [like Current] attached APPLICATION\nLine: 40\n create s.make\_from\_string (\"abcd\")\n-> asdas\n ensure\n\n-------------------------------------------------------------------------------",

"Has\_Compilation\_Error":true,

"Warning\_Message":"Warning code: Unused\_local\_warning\n\nWarning: unreferenced local variable(s)\nWhat to do: Remove it if you don't plan to use it in the future.\n\nClass: APPLICATION\nFeature: make\nUnused local is: \n\tc: INTEGER\_32\n\n-------------------------------------------------------------------------------",

"Has\_Warning":true,

"Needs\_Target":false,

"Dump\_Message":"Eiffel Compilation Manager\nVersion 14.05.9.5158 GPL Edition - win64\n\nDegree 6: Examining System\nDegree 5: Parsing Classes\nDegree 4: Analyzing Inheritance\nDegree 3: Checking Types\n-------------------------------------------------------------------------------\n\nWarning code: Unused\_local\_warning\n\nWarning: unreferenced local variable(s)\nWhat to do: Remove it if you don't plan to use it in the future.\n\nClass: APPLICATION\nFeature: make\nUnused local is: \n\tc: INTEGER\_32\n\n-------------------------------------------------------------------------------\n\nError code: VEEN\n\nError: unknown identifier.\nWhat to do: make sure that identifier, if needed, is final name of\n feature of class, or local entity or formal argument of routine.\n\nClass: APPLICATION\nFeature: extra\_feature\nIdentifier: asdas\nTarget type: [like Current] attached APPLICATION\nLine: 40\n create s.make\_from\_string (\"abcd\")\n-> asdas\n ensure\n\n-------------------------------------------------------------------------------\n",

"Error":[{

"Error\_Code":"VEEN\n",

"Error":"unknown identifier.\n",

"What\_to\_do":"make sure that identifier, if needed, is final name of\n feature of class, or local entity or formal argument of routine.\n\n",

"Class":"APPLICATION\n",

"Feature":"extra\_feature\n",

"Line":40,

"Before\_Line":"Identifier: asdas\nTarget type: [like Current] attached APPLICATION\n",

"After\_Line":" create s.make\_from\_string (\"abcd\")\n-> asdas\n ensure\n\n",

"Dump":"Error code: VEEN\n\nError: unknown identifier.\nWhat to do: make sure that identifier, if needed, is final name of\n feature of class, or local entity or formal argument of routine.\n\nClass: APPLICATION\nFeature: extra\_feature\nIdentifier: asdas\nTarget type: [like Current] attached APPLICATION\nLine: 40\n create s.make\_from\_string (\"abcd\")\n-> asdas\n ensure\n\n"}],

"Warning":[{

"Warning\_Code":"Unused\_local\_warning\n",

"Warning":"unreferenced local variable(s)\n",

"What\_to\_do":"Remove it if you don't plan to use it in the future.\n\n",

"Class":"APPLICATION\n",

"Feature":"make\n",

"After\_Feature":"Unused local is: \n\tc: INTEGER\_32\n\n",

"Dump":"Warning code: Unused\_local\_warning\n\nWarning: unreferenced local variable(s)\nWhat to do: Remove it if you don't plan to use it in the future.\n\nClass: APPLICATION\nFeature: make\nUnused local is: \n\tc: INTEGER\_32\n\n"

}]}],

"Targets":null,

"Compilation\_Succeeded":false

}]

**Note 1 :** After the server is started, it needs to set the project directory and location once for any new project(by the path flag in the compile parameters). If however the server is terminated while working, it will need the path location again for the project it was working on. So, the server must run continuously.

**Note 2 :** The server should always be started before the project is opened (i.e. compiled, otherwise it gives “You don’t have the permission to write to its EIFGENS’s” error)

**Execution**

<http://localhost:9090/run?id=id-given>

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter Name | Optional | Default Value (if optional) | Type |
| id | No |  | String |

When you run a project, you must always supply the project id in the URL. The server already has the path information stored with it so it does not need it.

The JSON response object for error free execution looks like this:

[{

"Execution\_Output":"Hello Eiffel World!\nNumber not out of range\n0",

"Error\_Message":"",

"Warning\_Message":"",

"Has\_Warning":false,

"Compile\_Errors":null,

"Has\_Compilation\_Error":false,

"Warnings”:null,

“Runtime\_Text”:””,

"Runtime\_Errors":null,

"Has\_Runtime\_Error":false,

"Execution\_Succeeded":true,

}]

The keys in the JSON response for execution

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Key | Type | Description | Default Value |
| Execution\_Output | String | Gives the execution output | “” |
| Error\_Message | String | Gives the unparsed error message (runtime error message if there is a runtime error or compile time error message if there is a compile time error) | “” |
| Warning\_Message | String | Gives the unparsed warning message | “” |
| Has\_Warning | Boolean | Indicates whether there is a warning message in the compilation. | True/false |
| Compile\_Errors | Json Array | Gives the parsed JSON object for the compile time error message with its keys | Null |
| Has\_Compilation\_Error | Boolean | Indicates whether there is a compilation error or not | True/false |
| Warnings | Json Array | Gives the parsed JSON object for the warning message with its keys | Null |
| Runtime\_Text | String | Gives the heading and message at the top of the table of the stack trace | “” |
| Runtime\_Errors | Json Array | Gives the parsed JSON object for the runtime error message with its keys | Null |
| Has\_Runtime\_Error | Boolean | Indicates whether there is a runtime error or not | True/false |
| Execution\_Succeeded | Boolean | Indicates whether the execution was successful or not.  The execution fails under the following circumstances:   1. Compilation Error : You can check the Has\_Compilation\_Error field, which will be true. 2. Runtime Error: You can check the Has\_Runtime\_Error field, which will be true. 3. Timeout: You can check the status code for timeouts (504). |  |

The keys for the Runtime\_Errors in the JSON response for execution

|  |  |  |
| --- | --- | --- |
| Name of Key | Type | Description |
| Class | String | Gives the class name |
| Feature | String | Gives the feature name |
| Routine | String | Gives the position in the feature of the class where the runtime exception occured |
| Message | String | Gives the message of the runtime error |
| Effect | String | Gives the effect of the runtime exception |

The above JSON-OBJECT is repeated for every stack trace, thereby making the Runtime\_Error JSON\_ARRAY.

An Example of a run time error message

[{

"Execution\_Output":"Hello Eiffel World!\nNumber not out of range\n0",

"Error\_Message":"\nsample: system execution failed.\nFollowing is the set of recorded exceptions:\n\n-------------------------------------------------------------------------------\nClass / Object Routine Nature of exception Effect\n-------------------------------------------------------------------------------\nACCOUNT \_invariant @2 balance\_positive: \n<00000056A6366A28> Class invariant violated. Fail\n-------------------------------------------------------------------------------\nACCOUNT \_invariant \n<00000056A6366A28> Routine failure. Fail\n-------------------------------------------------------------------------------\nACCOUNT withdraw @3 \n<00000056A6366A28> Routine failure. Fail\n-------------------------------------------------------------------------------\nAPPLICATION make @5 \n<00000056A6366588> Routine failure. Fail\n-------------------------------------------------------------------------------\nAPPLICATION root's creation \n<00000056A6366588> Routine failure. Exit\n-------------------------------------------------------------------------------\n",

"Warning\_Message":"",

"Has\_Warning":false,

"Compile\_Errors":null,

"Has\_Compilation\_Error":false,

"Warnings":null,

"Runtime\_Text":"\nsample: system execution failed.\nFollowing is the set of recorded exceptions:\n",

"Runtime\_Errors":[{

"Class":"ACCOUNT<00000056A6366A28>",

"Feature":"\_invariant",

"Routine":"2",

"Message":"balance\_positive:Class invariant violated.",

"Effect":"Fail"},

{"Class":"ACCOUNT<00000056A6366A28>",

"Feature":"\_invariant",

"Routine":"",

"Message":"Routine failure.",

"Effect":"Fail”},

{"Class":"ACCOUNT<00000056A6366A28>",

"Feature":"withdraw",

"Routine":"3",

"Message":"Routine failure.",

"Effect":"Fail"},

{"Class":"APPLICATION<00000056A6366588>",

"Feature":"make",

"Routine":"5",

"Message":"Routine failure.",

"Effect":"Fail”},

{"Class":"APPLICATION<00000056A6366588>",

"Feature":"root's creation",

"Routine":"",

"Message":"Routine failure.",

"Effect":"Exit"

}],

"Has\_Runtime\_Error":true,

"Execution\_Succeeded":false }]

In addition to the above response, a web socket is opened every time the user wants to execute his program. So, the web socket delivers a real time output to the user in the browser, and as soon as the execution is over, the server returns the response object as above, and then closes the websocket.

The Websocket is hearing on a different port 9999, so the client must connect on that port.

If the user wants to input something while execution, the following sequence of steps take place:

* If the output requires an input, it waits
* The user sends the input over the web socket connection.
* The server receives the input and feeds it into the process stream, and continues with the execution
* If the user does not give an input, the server waits for runtime\_timeout seconds, and then times out, and returns a 504 timeout status code.

**Flat View**

<http://localhost:9090/flatView?id=id-given>; class=your\_class

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter Name | Optional | Default Value (if optional) | Type |
| id | No |  | String |
| class | No |  | String |

When you want the flat view of a project, you must always supply the ‘id’ and the ‘class’ in the URL. The server already has the path information stored with it so it does not need it.

The JSON response object JSON for flat\_view looks like this:

[{

"Flat\_View":"………………………………………….",

"Error\_Message":"",

"Has\_Compilation\_Error":false,

"Has\_Flat\_View":true,

"Warning\_Message":"",

"Has\_Warning":true,

"Dump":"Eiffel Compilation Manager\nVersion 14.05.9.5158 GPL Edition - win64\n\nDegree 6: Examining System\nDegree 5: Parsing Classes\nDegree 4: Analyzing Inheritance\nDegree 3: Checking Types\nDegree 2: Generating Byte Code\nDegree 1: Generating Metadata\nMelting System Changes\n-------------------------------------------------------------------------------\n\nWarning code: Unused\_local\_warning\n\nWarning: unreferenced local variable(s)\nWhat to do: Remove it if you don't plan to use it in the future.\n\nClass: APPLICATION\nFeature: make\nUnused local is: \n\tc: INTEGER\_32\n\n-------------------------------------------------------------------------------\nSystem Recompiled.\nnote\ Flat\_View here",

"Errors":null,

"Warnings":[{

"Warning\_Code":"Unused\_local\_warning\n",

"Warning":"unreferenced local variable(s)\n",

"What\_to\_do":"Remove it if you don't plan to use it in the future.\n\n",

"Class":"APPLICATION\n",

"Feature":"make\n",

"After\_Feature":"Unused local is: \n\tc: INTEGER\_32\n\n",

"Dump":"Warning code: Unused\_local\_warning\n\nWarning: unreferenced local variable(s)\nWhat to do: Remove it if you don't plan to use it in the future.\n\nClass: APPLICATION\nFeature: make\nUnused local is: \n\tc: INTEGER\_32\n\n"}]}]

The keys in the JSON response for flat view

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Key | Type | Description | Default Values |
| Flat\_View | String | Gives the Flat View of the class | “” |
| Error\_Message | String | Gives the unparsed error message | “” |
| Has\_Compilation\_Error | Boolean | Indicates whether there is an error or not in the compilation | True/false |
| Has\_Flat\_View | Boolean | Indicates whether the response object contains the flat view or not.  The flat view is not shown under the following circumstances:   1. Compilation Error : You can check the Has\_Compilation\_Error field, which will be true. 2. Class does not exist: In this case Has\_Compilation\_Error is false, but the error message is non-empty and contains the error message for the class does not exist 3. Timeout: You can check the status code for timeouts (504) | True/False |
| Warning\_Message | String | Gives the unparsed warning message | “” |
| Has\_Warning | Boolean | Indicates whether there is a warning message in the compilation. | True/false |
| Dump | String | Gives the entire dump message with errors, warnings, and the flat view. | “” |
| Errors | Json Array | Gives the parsed JSON object for the error message with its keys | Null |
| Warnings | Json Array | Gives the parsed JSON object for the warning message with its keys | Null |

**Contract View**

<http://localhost:9090/contractView?id=id-given>; class=your\_class

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter Name | Optional | Default Value (if optional) | Type |
| id | No |  | String |
| class | No |  | String |

When you want the contract view of a project, you must always supply the ‘id’ and the ‘class’ in the URL. The server already has the path information stored with it so it does not need it.

The JSON response object JSON for contract\_view looks like this:

[{

"Contract\_View":"note\n\tdescription: \"Summary description for {ACCOUNT}.\"\n\tauthor: \"\"\n\tdate: \"$Date$\"\n\trevision: \"$Revision$\"\n\nclass interface\n\tACCOUNT\n\ncreate \n\tmake\n\nfeature \n\n\tbalance: REAL\_64\n\n\tdeposit (amt: REAL\_64)\n\t\trequire\n\t\t\tamt\_positive: amt > 0.to\_double\n\t\tensure\n\t\t\t\tbalance = old balance + amt\n\n\tmake (a: REAL\_64)\n\t\trequire\n\t\t\tbal\_positive: a > 0.to\_double\n\t\tensure\n\t\t\tbal\_set: balance = a\n\n\twithdraw (amt: REAL\_64)\n\t\trequire\n\t\t\tamt\_positive: amt > 0.to\_double\n\t\tensure\n\t\t\t\tbalance = old balance - amt\n\t\ninvariant\n\tbalance\_positive: balance > 0.to\_double\n\nend -- class ACCOUNT\n\n",

"Error\_Message":"",

"Has\_Compilation\_Error":false,

"Has\_Contract\_View":true,

"Warning\_Message":"",

"Has\_Warning":false,

"Dump":"Eiffel Compilation Manager\nVersion 14.05.9.5158 GPL Edition - win64\n\nDegree 6: Examining System\nSystem Recompiled.\nnote\n\tdescription: \"Summary description for {ACCOUNT}.\"\n\tauthor: \"\"\n\tdate: \"$Date$\"\n\trevision: \"$Revision$\"\n\nclass interface\n\tACCOUNT\n\ncreate \n\tmake\n\nfeature \n\n\tbalance: REAL\_64\n\n\tdeposit (amt: REAL\_64)\n\t\trequire\n\t\t\tamt\_positive: amt > 0.to\_double\n\t\tensure\n\t\t\t\tbalance = old balance + amt\n\n\tmake (a: REAL\_64)\n\t\trequire\n\t\t\tbal\_positive: a > 0.to\_double\n\t\tensure\n\t\t\tbal\_set: balance = a\n\n\twithdraw (amt: REAL\_64)\n\t\trequire\n\t\t\tamt\_positive: amt > 0.to\_double\n\t\tensure\n\t\t\t\tbalance = old balance - amt\n\t\ninvariant\n\tbalance\_positive: balance > 0.to\_double\n\nend -- class ACCOUNT\n\n",

"Errors":null,

"Warnings":null}]

The keys in the JSON response for flat view

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Key | Type | Description | Default Values |
| Contract\_View | String | Gives the Contract View of the class | “” |
| Error\_Message | String | Gives the unparsed error message | “” |
| Has\_Compilation\_Error | Boolean | Indicates whether there is an error or not in the compilation | True/false |
| Has\_Contract\_View | Boolean | Indicates whether the response object contains the contract view or not.  The contract view is not shown under the following circumstances:   1. Compilation Error : You can check the Has\_Compilation\_Error field, which will be true. 2. Class does not exist: In this case Has\_Compilation\_Error is false, but the error message is non-empty and contains the error message for the class does not exist 3. Timeout: You can check the status code for timeouts (504) | True/False |
| Warning\_Message | String | Gives the unparsed warning message | “” |
| Has\_Warning | Boolean | Indicates whether there is a warning message in the compilation. | True/false |
| Dump | String | Gives the entire dump message with errors, warnings, and the flat view. | “” |
| Errors | Json Array | Gives the parsed JSON object for the error message with its keys | Null |
| Warnings | Json Array | Gives the parsed JSON object for the warning message with its keys | Null |

**Class Descendants**

<http://localhost:9090/classDescendants?id=id-given>;class=your\_class

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter Name | Optional | Default Value (if optional) | Type |
| id | No |  | String |
| class | No |  | String |

When you want the descendants of a class, you must always supply the ‘id’ and the ‘class’ in the URL. The server already has the path information stored with it so it does not need it.

The response object JSON for classDescendants looks like this:

[{

"Class\_Descendants\_Dump":"\n\tAPPLICATION\n\n",

"Error\_Message":"",

"Has\_Compilation\_Error":false,

"Has\_Descendants":true,

"Warning\_Message":"",

"Has\_Warning":false,

"Dump":"Eiffel Compilation Manager\nVersion 14.05.9.5158 GPL Edition - win64\n\nDegree 6: Examining System\nSystem Recompiled.\n\n\tAPPLICATION\n\n",

"Errors":null,

"Warnings":null,

"Descendants":[{"Deferred":false,"Class\_Name":"APPLICATION","Descendants":[{"Deferred":false,"Class\_Name":"Class\_A","Descendants":[]},{"Deferred":false,"Class\_Name":"Class\_B","Descendants":[]}]

}]

}]

The keys in the JSON response for class descendants

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Key | Type | Description | Default Values |
| Class\_Descendants\_Dump | String | Gives the dump of the Descendants of the class | “” |
| Error\_Message | String | Gives the unparsed error message | “” |
| Has\_Compilation\_Error | Boolean | Indicates whether there is an error or not in the compilation | True/false |
| Has\_Descendants | Boolean | Indicates whether the response object contains the descendants or not  The descendants is not shown under the following circumstances:   1. Compilation Error : You can check the Has\_Compilation\_Error field, which will be true. 2. Class does not exist: In this case Has\_Compilation\_Error is false, but the error message is non-empty and contains the error message for the class does not exist 3. Timeout: You can check the status code for timeouts (504) | True/False |
| Warning\_Message | String | Gives the unparsed warning message | “” |
| Has\_Warning | Boolean | Indicates whether there is a warning message in the compilation. | True/false |
| Dump | String | Gives the entire dump message with errors, warnings, and the flat view. | “” |
| Errors | Json Array | Gives the parsed JSON object for the error message with its keys | Null |
| Warnings | Json Array | Gives the parsed JSON object for the warning message with its keys | Null |
| Descendants | Json Array | Gives the JSON array for the parsed descendants structure | Null |

The keys in the Descendants JSON array

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Key | Type | Description | Default Values |
| Class\_Name | String | Gives the Class Name | “” |
| Deferred | Boolean | Indicates whether the class is a deferred class or not | True/False |
| Descendants | Json Array | Gives the descendants of this class | Null |

**Class Ancestors**

<http://localhost:9090/classAncestors?id=id-given>;class=your\_class

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter Name | Optional | Default Value (if optional) | Type |
| id | No |  | String |
| class | No |  | String |

When you want the ancestors of a class, you must always supply the ‘id’ and the ‘class’ in the URL. The server already has the path information stored with it so it does not need it.

The response object JSON for class Ancestors looks like this:

[{

"Class\_Ancestors\_Dump":"\n\tAPPLICATION\n\n",

"Error\_Message":"",

"Has\_Compilation\_Error":false,

"Has\_Ancestors":true,

"Warning\_Message":"",

"Has\_Warning":false,

"Dump":"Eiffel Compilation Manager\nVersion 14.05.9.5158 GPL Edition - win64\n\nDegree 6: Examining System\nSystem Recompiled.\n\n\tAPPLICATION\n\n",

"Errors":null,

"Warnings":null,

"Ancestors":[{"Deferred":false,"Class\_Name":"APPLICATION","Ancestors":[{"Deferred":false,"Class\_Name":"Class\_A","Ancestors":[]},{"Deferred":false,"Class\_Name":"Class\_B","Ancestors":[]}]

}]

}]

The keys in the JSON response for class ancestors

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Key | Type | Description | Default Values |
| Class\_Ancestors\_Dump | String | Gives the dump of the Ancestors of the class | “” |
| Error\_Message | String | Gives the unparsed error message | “” |
| Has\_Compilation\_Error | Boolean | Indicates whether there is an error or not in the compilation | True/false |
| Has\_Ancestors | Boolean | Indicates whether the response object contains the ancestors or not.  The ancestors is not shown under the following circumstances:   1. Compilation Error : You can check the Has\_Compilation\_Error field, which will be true. 2. Class does not exist: In this case Has\_Compilation\_Error is false, but the error message is non-empty and contains the error message for the class does not exist 3. Timeout: You can check the status code for timeouts (504) | True/False |
| Warning\_Message | String | Gives the unparsed warning message | “” |
| Has\_Warning | Boolean | Indicates whether there is a warning message in the compilation. | True/false |
| Dump | String | Gives the entire dump message with errors, warnings, and the flat view. | “” |
| Errors | Json Array | Gives the parsed JSON object for the error message with its keys | Null |
| Warnings | Json Array | Gives the parsed JSON object for the warning message with its keys | Null |
| Ancestors | Json Array | Gives the JSON array for the parsed ancestors structure | Null |

The keys in the Ancestors JSON array

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Key | Type | Description | Default Values |
| Class\_Name | String | Gives the Class Name | “” |
| Deferred | Boolean | Indicates whether the class is a deferred class or not | True/False |
| Ancestors | Json Array | Gives the ancestors of this class | Null |

**Class Clients**

<http://localhost:9090/classClients?id=id-given>;class=your\_class

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter Name | Optional | Default Value (if optional) | Type |
| id | No |  | String |
| class | No |  | String |

When you want the clients of a class, you must always supply the ‘id’ and the ‘class’ in the URL. The server already has the path information stored with it so it does not need it.

The response object JSON for class clients looks like this:

[{

"Class\_Clients\_Dump":"\n\tAPPLICATION\n\n",

"Error\_Message":"",

"Has\_Compilation\_Error":false,

"Has\_Clients":true,

"Warning\_Message":"",

"Has\_Warning":false,

"Dump":"Eiffel Compilation Manager\nVersion 14.05.9.5158 GPL Edition - win64\n\nDegree 6: Examining System\nSystem Recompiled.\n\n\tAPPLICATION\n\n",

"Errors":null,

"Warnings":null,

"Clients":[{"Deferred":false,"Class\_Name":"SampleA},{"Deferred":false,"Class\_Name":"SampleB”}]

}]

The keys in the JSON response for class clients

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Key | Type | Description | Default Values |
| Class\_Clients\_Dump | String | Gives the dump of the clients of the class | “” |
| Error\_Message | String | Gives the unparsed error message | “” |
| Has\_Compilation\_Error | Boolean | Indicates whether there is an error or not in the compilation | True/false |
| Has\_Clients | Boolean | Indicates whether the response object contains the clients or not.  The clients is not shown under the following circumstances:   1. Compilation Error : You can check the Has\_Compilation\_Error field, which will be true. 2. Class does not exist: In this case Has\_Compilation\_Error is false, but the error message is non-empty and contains the error message for the class does not exist 3. Timeout: You can check the status code for timeouts (504) | True/False |
| Warning\_Message | String | Gives the unparsed warning message | “” |
| Has\_Warning | Boolean | Indicates whether there is a warning message in the compilation. | True/false |
| Dump | String | Gives the entire dump message with errors, warnings, and the flat view. | “” |
| Errors | Json Array | Gives the parsed JSON object for the error message with its keys | Null |
| Warnings | Json Array | Gives the parsed JSON object for the warning message with its keys | Null |
| Clients | Json Array | Gives the JSON array for the parsed clients structure | Null |

The keys in the Clients JSON array

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Key | Type | Description | Default Values |
| Class\_Name | String | Gives the Class Name | “” |
| Deferred | Boolean | Indicates whether the class is a deferred class or not | True/False |

**Class Suppliers**

<http://localhost:9090/classSuppliers?id=id-given>;class=your\_class

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter Name | Optional | Default Value (if optional) | Type |
| id | No |  | String |
| class | No |  | String |

When you want the suppliers of a class, you must always supply the ‘id’ and the ‘class’ in the URL. The server already has the path information stored with it so it does not need it.

The response object JSON for class supplier looks like this:

[{

"Class\_Suppliers\_Dump":"\n\tAPPLICATION\n\n",

"Error\_Message":"",

"Has\_Compilation\_Error":false,

"Has\_Suppliers":true,

"Warning\_Message":"",

"Has\_Warning":false,

"Dump":"Eiffel Compilation Manager\nVersion 14.05.9.5158 GPL Edition - win64\n\nDegree 6: Examining System\nSystem Recompiled.\n\n\tAPPLICATION\n\n",

"Errors":null,

"Warnings":null,

"Supliers":[{"Deferred":false,"Class\_Name":"SampleA},{"Deferred":false,"Class\_Name":"SampleB”}]

}]

The keys in the JSON response for class clients

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Key | Type | Description | Default Values |
| Class\_Suppliers\_Dump | String | Gives the dump of the suppliers of the class | “” |
| Error\_Message | String | Gives the unparsed error message | “” |
| Has\_Compilation\_Error | Boolean | Indicates whether there is an error or not in the compilation | True/false |
| Has\_Suppliers | Boolean | Indicates whether the response object contains the suppliers or not.  The suppliers is not shown under the following circumstances:   1. Compilation Error : You can check the Has\_Compilation\_Error field, which will be true. 2. Class does not exist: In this case Has\_Compilation\_Error is false, but the error message is non-empty and contains the error message for the class does not exist 3. Timeout: You can check the status code for timeouts (504) | True/False |
| Warning\_Message | String | Gives the unparsed warning message | “” |
| Has\_Warning | Boolean | Indicates whether there is a warning message in the compilation. | True/false |
| Dump | String | Gives the entire dump message with errors, warnings, and the flat view. | “” |
| Errors | Json Array | Gives the parsed JSON object for the error message with its keys | Null |
| Warnings | Json Array | Gives the parsed JSON object for the warning message with its keys | Null |
| Suppliers | Json Array | Gives the JSON array for the parsed suppliers structure | Null |

The keys in the Suppliers JSON array

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Key | Type | Description | Default Values |
| Class\_Name | String | Gives the Class Name | “” |
| Deferred | Boolean | Indicates whether the class is a deferred class or not | True/False |

**Feature Callers**

<http://localhost:9090/featureCallers?id=id-given>;class=your\_class;feature=your\_feature

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter Name | Optional | Default Value (if optional) | Type |
| id | No |  | String |
| class | No |  | String |
| Feature | No |  | String |

When you want the feature callers of a feature of a class, you must always supply the ‘id’, ‘class’ and ‘feature’ in the URL. The server already has the path information stored with it so it does not need it.

The response object JSON for feature caller looks like this:

[{

"Feature\_Callers\_Dump":"balance from ACCOUNT\n\tACCOUNT\n\t\tdeposit\n\t\tinvariant\n\t\tmake\n\t\twithdraw\n\n",

"Error\_Message":"",

"Has\_Compilation\_Error":false,

"Has\_Feature\_Callers":true,

"Warning\_Message":"",

"Has\_Warning":false,

"Dump":"Eiffel Compilation Manager\nVersion 14.05.9.5158 GPL Edition - win64\n\nDegree 6: Examining System\nSystem Recompiled.\nbalance from ACCOUNT\n\tACCOUNT\n\t\tdeposit\n\t\tinvariant\n\t\tmake\n\t\twithdraw\n\n",

"Errors":null,

"Warnings":null,

"Callers":[{"Class\_Name":"ACCOUNT","Features":[{"Feature\_Name":"deposit"},{"Feature\_Name":"invariant"},{"Feature\_Name":"make"},{"Feature\_Name":"withdraw"}]}]

}]

The keys in the JSON response for class clients

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Key | Type | Description | Default Values |
| Feature\_Callers\_Dump | String | Gives the dump of the callers of the feature of the class | “” |
| Error\_Message | String | Gives the unparsed error message | “” |
| Has\_Compilation\_Error | Boolean | Indicates whether there is an error or not in the compilation | True/false |
| Has\_Feature\_Callers | Boolean | Indicates whether the response object contains the feature callers or not.  The feature callers is not shown under the following circumstances:   1. Compilation Error : You can check the Has\_Compilation\_Error field, which will be true. 2. Class/Feature does not exist: In this case Has\_Compilation\_Error is false, but the error message is non-empty and contains the error message for the class does not exist 3. Timeout: You can check the status code for timeouts (504) | True/False |
| Warning\_Message | String | Gives the unparsed warning message | “” |
| Has\_Warning | Boolean | Indicates whether there is a warning message in the compilation. | True/false |
| Dump | String | Gives the entire dump message with errors, warnings, and the flat view. | “” |
| Errors | Json Array | Gives the parsed JSON object for the error message with its keys | Null |
| Warnings | Json Array | Gives the parsed JSON object for the warning message with its keys | Null |
| Callers | Json Array | Gives the JSON array for the parsed callers structure | Null |

The keys in the Callers JSON array

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Key | Type | Description | Default Values |
| Class\_Name | String | Gives the Class Name | “” |
| Features | Json Array | Gives the list of features of Class\_Name that call the requested feature | Null |

The keys in the Features JSON array

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Key | Type | Description | Default Values |
| Feature\_Name | String | Gives the Feature Name | “” |

**Command Line**

<http://localhost:9090/commandLine?id=id-given>;command\_line=your\_message

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter Name | Optional | Default Value (if optional) | Type |
| id | No |  | String |
| command\_line | No |  | String |

When you want to provide a command line argument, you must always supply the ‘id’and the ‘command\_line’ in the URL. The server already has the path information stored with it so it does not need it.

The response object JSON for command line output for the command\_line=”ec –config sample.ecf –callers ACCOUNT balance” looks like this:

[{

"Command\_Line\_Dump":"balance from ACCOUNT\n\tACCOUNT\n\t\tdeposit\n\t\tinvariant\n\t\tmake\n\t\twithdraw\n\n

}]

The keys in the JSON response for class clients

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Key | Type | Description | Default Values |
| Command\_Line\_Dump | String | Gives the dump of the output from the compiler after executing the command line argument given by the user | “” |

**Compilation and Runtime timeout**

The server would respond with a 504 Gateway Error and the normal response body, if the compilation or execution time exceeded the values set for the respective timeouts.

Inside ./www, is a file config.json

Change the Compilation\_Timeout and Runtime\_Timeout values (in seconds) for your need.

Currently the keys and their values are

|  |  |  |
| --- | --- | --- |
| Key | Value (in milliseconds) | Type |
| Compilation\_Timeout | 180000 | Integer |
| Runtime\_Timeout | 10000 | Integer |

An Example of execution timeout response from the server is:

{

"data":[{

"Execution\_Output":"Hello Eiffel World!\r\nNumber not out of range\r\n000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000",

"Error\_Message":"",

"Warning\_Message":"",

"Has\_Warning":false,

"Compile\_Errors":null,

"Has\_Compilation\_Error":false,

"Warnings":null,

"Runtime\_Text":"",

"Runtime\_Errors":null,

"Has\_Runtime\_Error":false,

"Execution\_Succeeded":false}],

"status":504,

"config":{

"method":"GET",

"transformRequest":[null],

"transformResponse":[null],

"url":"/run",

"params":{"id":"90799D1A-C197-4822-BEE9-C0D076CA8562"},

"headers":{"Accept":"application/json, text/plain, \*/\*"}

}

}

**The server returns the above response when a 504 is occurred.**

The keys in the JSON response for 504 from the server is

|  |  |  |  |
| --- | --- | --- | --- |
| Name of Key | Type | Description | Default Values |
| data | Json Array | Contains the JSON array for the corresponding function (eg, execution in this case).  The Execution\_Output(in runtime\_timeout) and Compile\_Message(in compilation\_timeout) are displayed till the point they were not timed out. | null |
| status | Integer | Gives the status Code | 504 (always in our case) |
| config | Json Object | Gives the method, url, transformRequest, transformResponse, params, and headers. |  |

**Response Status Codes**

|  |  |
| --- | --- |
| Status Code | Description |
| 200 | Ok |
| 410 | Gone – The user tried to access a project that either does not exist or has been inactive for too long and is ready for deletion. |
| 504 | Gateway Timeout – The server did not get a response from the upstream server on time. Represents a compilation or a runtime timeout. |

**Removing Old Projects**

The server creates a new thread that runs continuously. It checks after a periodic interval of time (fixed frequency), if the user has not used a particular project for a fixed long time (say 90 days). If it is true, it deletes the project to save memory.

The developer can set these fixed values by changing them in the config.json file inside ./www folder.

Currently the keys and their values are

|  |  |  |  |
| --- | --- | --- | --- |
| Key | Value | Type | Description |
| Periodic\_Time | 25000(in milliseconds) | Integer | It defines the time interval(in milliseconds) after which the thread checking for old projects should repeat itself. |
| Project\_Duration | 240000(in seconds) | Integer | It defines the time(in seconds) after which a user project is considered inactive, and can be removed by the server. |
| Pre\_Project\_Duration | 300000(in seconds) | Integer | It defines the time(in seconds) after which a user project is considered inactive. The project is not yet deleted, but the user cannot access it, and the server returns a 410 (Gone) response. |
| Initial\_Sleep\_Time | 3000(in milliseconds) | Integer | The server when launched, halts everything for Initial\_Sleep\_Time(in milliseconds), and deletes the old projects in the meantime. |
| New\_Project\_Wait\_Time | 120000000000(in nanoseconds) | Integer\_64 | The server will make the deletion thread sleep for this time whenever a new project is created. This will give the server the time to update its ‘date-project’ table with this new project. |

The server stops the user from using a project if it is inactive for more than Pre\_Project\_Duration time, and returns a 410(Gone) status code with the response if the user tries to access it afterwards.   
But the project is not yet deleted. The child thread deletes the project after Project\_Duration time. This ensures that a race condition never occurs, as the user is restricted from accessing the project a little before the deletion starts (t1: Time the user can access to the project, t2: Lifetime of the project; we always ensure t1<t2, so a race condition never occurs).

**Apache Configuration**

Put the final executable file in the /htdocs folder of apache. The name of the executable should be ‘name\_libfcgi.exe’. Also, put libfcgi.dll and libfcgi.lib in the same place. These 2 files can be found at $ISE\_LIBRARY\contrib\library\web\framework\ewf\libfcgi\spec. Now we have to modify the httpd.conf file, inside apache2.2/conf folder.

Make sure the following modules are loaded:

LoadModule fcgid\_module modules/mod\_fcgid.so

LoadModule rewrite\_module modules/mod\_rewrite.so

Wherever you find AllowOverride None, change it to AllowOverride All.

Inside the <Directory “C:/Apache2.2/cgi-bin”> and the <Directory “C:/Apache2.2/htdocs”>,

add the following line: Options +ExecCGI +Includes +FollowSymLinks –Indexes

In the htdocs folder, add a .htaccess file with the following content (Replace app\_libfcgi.exe with the name of your executable below):

Options +ExecCGI +Includes +FollowSymLinks -Indexes

<IfModule mod\_fcgid.c>

AddHandler fcgid-script .ews

FcgidWrapper "C:/Apache2.2/htdocs/app\_libfcgi.exe" .ews

</IfModule>

<IfModule mod\_rewrite.c>

RewriteEngine on

RewriteBase /

RewriteRule ^$ service.ews [L]

RewriteCond %{REQUEST\_FILENAME} !­f

RewriteCond %{REQUEST\_FILENAME} !­d

RewriteCond %{REQUEST\_URI} !=/favicon.ico

RewriteCond %{REQUEST\_URI} !service.ews

RewriteRule ^(.\*)$ service.ews/$1

RewriteRule .\* - [E=HTTP\_AUTHORIZATION:%{HTTP:Authorization},L]

</IfModule>

Also, create a blank service.ews file in the same directory.

Make sure you also place the ‘projects’, ‘www’ and ‘test\_files’ folder in the same place as that of the executable.

You don’t need any index.html.

Just launch the apache server, and run it on localhost to test it. You can add the server details and configuration in httpd.conf for launching it on a live server.

**Assumptions**

* The developer will always give a valid non-empty user id. If the developer does not give a valid id, the response contains 410 status code (Gone).
* The developer will always give a path for a new project
* The developer will ensure that while a particular feature is in use, all the features are disabled from being used, till such time the feature use if over
* The developer will make sure that the end user should have a successful compilation with needs\_target=false, and then use the other features
* The developer will always give a class for the class functions
* The developer will always give a feature for the feature functions
* If any function from the API (except) compile/clean compile is called with a wrong\_target name/ no target\_name for a multi-target project, it will produce an error.