Design choices rationale:

In order to implement the security policy related to mobile code, **stack inspection is adopted**, where a set of flags is used to identify the presence on the call stack of potentially dangerous execution of mobile code with respect to private local code.

The used flags are: **Write | NoWrite | Read | NoRead | Send | NoSend | Exe**, this set extendible with couples of flags like **Do** and **DoNot**, where **Do** represent the flag associated to a potentially dangerous method and **DoNot** is the flag used to represent the protection of a data against the execution of method **Do**.

The choice of specifing for each network-based method the couple **Do - DoNot** was made in order to deal with mobile code in a more flexible way. Another possibility was to use only two flags: **Protected** for LocalCode and **Dangerous** for network-based methods but in this way the LocalCode could never be used inside a mobile code evaluation.

In fact, using pairs like **Do, DoNot** allows us to use our LocalCode inside mobile code execution and specify in which case we want the execution to be stopped.

e.g.:

myPin = LocalCode(Eint 12314, [NoWrite,NoRead,NoSend])
myAge = LocalCode(Eint 23, [NoWrite])

In this example we want myPin to be used only locally and never be exploited by methods like Write, Read, and Send executed by a mobile code. Otherwise, we want myAge to be used by mobilecode, the only operation that we want to deny is the overwriting of my age with the write method. (assuming that the write operation results in an assignment)

The only unpaired flag in the set is **Exe** who is associated to the **Execute** method. If and only if this flag is present on the permission stack the evaluation takes place inside the TEE. If Exe is not present inside the permission stack the expression is evaluated without considering, the security policies about mobile code.

This choice was made in such a way as to allow the execution of LocalCode without any trouble when it is exploited outside the mobile code context.

e.g.:

myAge = LocalCode(Eint 23, [NoWrite])

e1 = Execute(Write(myAge)
e2 = Write(myAge)

In this example the first expression will fail cause of security policy violation: we are trying to execute a mobile code that access the data myAge with a Write method. The second one will execute without troubles because we are locally using the method Write to override myAge.