

BI3 Specification

Secure isolated processes

BI3-TAE Emulator

The BI3-TAE resolves a number of the abstractions that are specified by the Theory of Absolute Information Security (TAIS) within the TAIS Alpha Environment (TAE). The BI3-TAE follows a general format that is reapplied for the other more advanced environments:

- **Circuit layout**

Each environment has a physical circuit layout that identifies each microprocessor and the various buses that connect it to other components. Even though these components are descriptions of a physical implementation, they are nonetheless an abstraction of a hardware manifestation. Many implementations will take a software form and map this layout into structures that will deliver an optimised

performance for the local architecture on which the simulation is presented.

- **Primitives**

Primitives are fundamental units of information that can be referenced on the CPU (within code) to access data on other microprocessors. All data within the BI3-TAE Emulator must be encoded within one of these primitive types, or be encoded within the TAE-RAM.

- **Factory load**

The factory loading state occurs once in a lifetime of the BI3-TAE emulator. At this point data is encoded into a deployment form. As the name implies, the Factory load would normally occur during the

construction of the BI3-TAE Emulator deployment. From the developer's perspective, it is likely that the BI3-TAE Emulator will be built to a state prior to the Factory load. This will allow developers to configure the emulator before running Factory load. Once the Factory load has been executed, the software becomes both optimised and tamper resistant, for example, the source code may be jettisoned and the binary encrypted.

- **Booting**

Each time the BI3-TAE Emulator is powered on, it initiates the boot sequence. The boot sequence can only take place after the Factory load is already

executed. The boot sequence prepares the BI3-TAE Emulator for normal operation and loads the various agents into memory and into their appropriate state.

- **Normal operation**

Once the boot sequence has completed, the BI3-TAE Emulator begins normal operation. For each TAIS environment, this may be slightly different but generally this will involve each microprocessor responding to asynchronous request from the CPU. The CPU itself is primarily concerned with switching between TAE processes.

- **Shutdown**

Shutdown occurs when an appropriate trigger for the TAIS environment is invoked. This shutdown sequence will prepare the BI3-TAE Emulator for a successful reboot.

- **Interfaces**

The major programming class interfaces are listed in a universal language independent format.

- **Special**

Each environment has a number of unique characteristics that emerge from the various microprocessors.

Resolving criteria of abstraction

The Theory of Absolute Information Security (TAIS) is a general theory of security and as such has a number of criteria of abstraction. These are broad design criteria which must hold true in the final deployment. However, TAIS deliberately leaves these abstract wherever possible, so as to maximise the freedom of design. The BI3 is a specification (as opposed to a theory) and as such, locks down many of the abstractions presented by TAIS. TAIS is designed in such a way as to allow for the majority of criteria to be locked

down by the manufacturers, thereby reducing the overall risk for the IT administrator. These manufacturing criteria are known as the Design-Time Factors. The administrator is only expected to manage a minimal set of factors known as the Run-Time Factors.

For any deployment, all the criteria must be ultimately resolved. This is very similar to Object Orientation that must resolve all abstract classes before execution can begin. Resolution is essential because the final

deployment must be able to determine its ultimate form so that it can manifest itself on physical hardware running precise software. Therefore, once a single *criterion of abstraction* has been resolved, it no longer needs to be considered. It can neither become unresolved again, nor can it be resolved twice in a different manner. Resolution to the criteria of abstraction is tracked in a table:

Resolution of criteria of abstraction						
Factors		Design-Time Factors				Run-Time Factors
TAE Factor	Factor summary	Resolved in TAIS	Resolved in BI3	Resolved in implementation	Resolved in substrate	Resolved in configuration

The table lists the stages that the implementation will move from in its most abstract to its most concrete form, from left to right, these are:

1. TAIS

TAIS resolves the broadest criteria such as the type of encapsulating mathematics used (Set Theory) and the architecture of the grid network

2. B13

The B13 resolves the layout of the microprocessors, buses, primitive types and defines an abstract machine operation model

3. The implementation

From the B13 perspective, the implementation is not specified, however, it is required to resolve the actual machine instruction model. It also includes the B13 manifestation which could be either hardware (as per the B13) or software emulation.

4. The substrate (if present)

In the case of software emulation, a substrate will be required. The substrate is likely to be an Operating

System that is bulked with the chosen hardware platform.

5. The local configuration

The Run-Time Factors present themselves as the local configuration and is the responsibility of the IT department and administrator. They include selecting appropriate devices, assigning people and information to Infosopes, choosing peers on the network and measuring the resulting security profile for each Infosope.

The grey cells indicate resolved criteria. Moving from left to right, resolution will always take the form of:

No	No	No	Yes	N/A	N/A	N/A
----	----	----	-----	-----	-----	-----

This means that once a criterion is resolved (Yes), then the remaining stages to deployment will be *Not Applicable (N/A)* as the manifestation becomes less abstract and more concrete.

Universal code convention

All source code listed here is stated in a language neutral form:

Classes are written as:

```
public class MyClass()  
{  
  
}
```

Methods are called as:

```
public    whole returnValue1,  
         whole returnValue2,  
         whole returnValueN    methodName ( whole parameter1,  
                                           whole parameter2,  
                                           whole parameterN) ;
```

Circuit layout

The BI3-TAE Emulator is comprised physically with a CPU capable of processing some abstract and undefined instruction set at 64, 128 or 256 bits.

This means that the buses must be capable of transmitting these data sizes with each instruction.

The CPU may contain an onboard memory cache, but the architecture must emulate the physical layout of Figure 1.

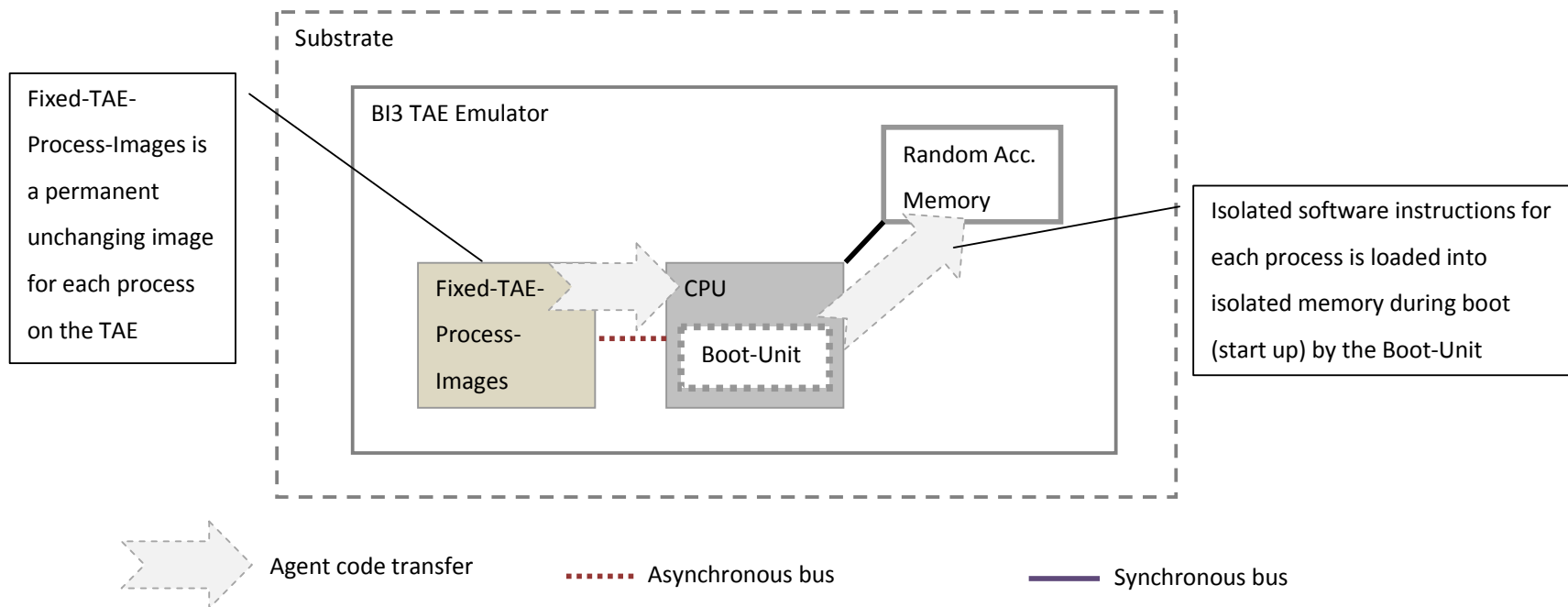


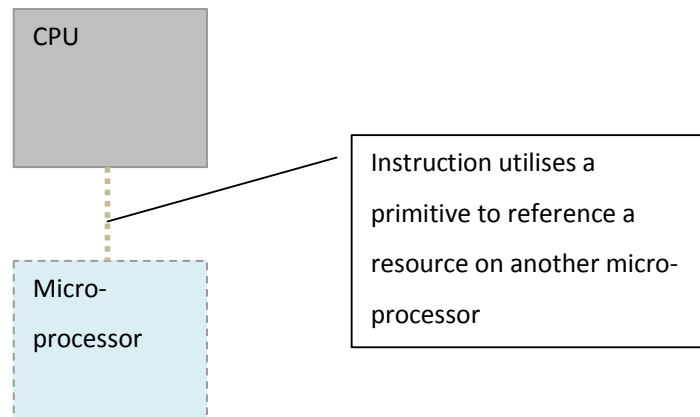
Figure 1 – The BI3-TAE Emulator is limited and probably only suited for isolated mathematical processes for which there are no other inputs

Resolution of criteria of abstraction for the BI3-TAE Emulator				
Factors		Design-Time Factors		
		Run-Time Factors		
TAE Factor ¹	Factor summary	Resolved in TAIS	Resolved in BI3	Nature of resolution
1@A1a	Substrate starts uncompromised	No	No	N/A
2@A1a	No software installed	No	No	N/A
3@A1a	No tampering	No	No	N/A
4@A1a	Identical restart	No	No	N/A
1@A1b	Isolation	No	No	N/A
1@A1c	No failure	No	No	N/A
1@A2a	Computationally complete	No	Yes	Basic von Neumann architecture
2@A2a	Multiple processes	No	No	N/A
3@A2a	Real-time scheduling	No	No	N/A
4@A2a	Singled threaded	No	No	N/A
5@A2a	TAE-RAM	No	No	N/A
6@A2a	Math	No	No	N/A
7@A2a	Undefined loading	No	Yes	Loaded from Fixed-Program-Images
1@A2b	Isolated memory	No	No	N/A
1@A2c	Prevent Malicious Client Problem	No	No	N/A
1@A2d	No boot parameters	No	Yes	The Fixed-TAE-Process-Images are read only
1@A2e	No library calls	No	No	N/A
1@A3	Store information in TAE process	No	No	N/A
2@A3	Access information not accounted for	No	No	N/A

¹ Reference “The 7 Steps of Absolute Information Security with TAIS, version 0.912”

Primitives

There are no primitives in the BI3 TAE Emulator because all instructions are committed by the CPU will access the TAE-RAM directly. The primitives are only required (in more advanced TAIS environments) by the CPU instructions themselves to access data that resides on another micro-processor and is accessed by a specific bus.



Factory loading

In a software implementation, Factory loading of the BI3-TAE Emulator would ordinarily occur the first time that the build is executed. Factory loading is an opportunity for the developers to load resources and embed them in a tamper resistant and efficient form.

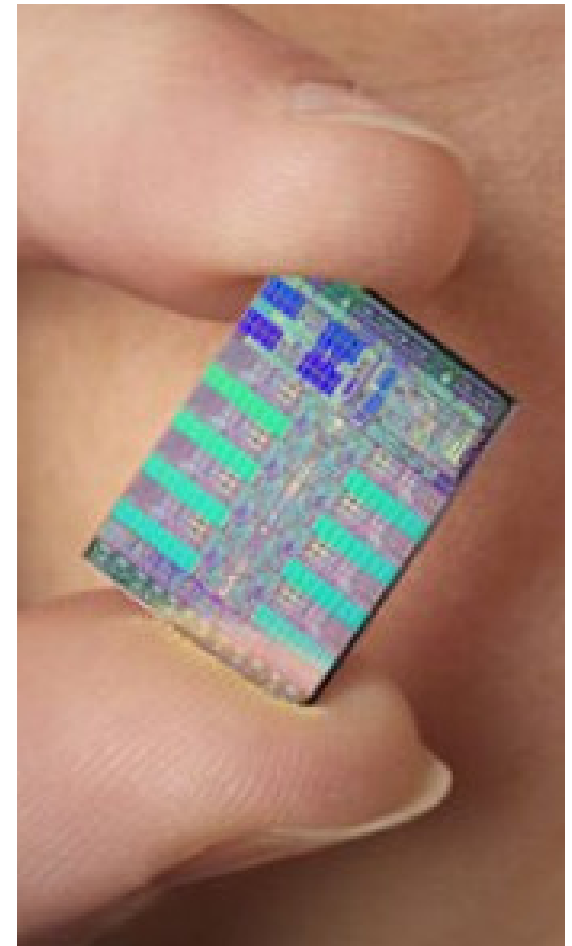
The Factory load is initiated by the BI3-TAE Emulator on first run which calls the `factoryLoad()` method of the `BI3TAE` class.

```
public void factoryLoad() ;
```

The Factory load executes in a privileged mode that allows the code to access other resources that would otherwise not be available on the 2nd and following runs. During the 1st run, resources are transferred and loaded into the Fixed-TAE-Process-Images (see Circuit layout).

The specifics of how the Factory load works is left undefined, so in an embedded implementation, the Fixed-Agent-Images may simply be preloaded when the board is assembled. In this case, the `factoryLoad()` method would never actually be called. This step is managed by the production process, so the resulting behaviour is equivalent (but not identical to) a software implementation for which it was called.

(Right) Hardware circuits may avoid a formal call to `factoryLoad()`



Booting

The BI3-TAE Emulator will start each time from the same state by adhering to the following sequence:

1. All microprocessors are powered.
2. All microprocessors send a signal to the CPU indicating that they function normally and have paused while they await further instruction from the CPU (on their bus) before commencing.
3. The Boot-Unit instructs the CPU to load data from the Fixed-TAE-Process-Images to set up the TAE-RAM for each TAE process image in order for it to begin executing.
4. The CPU signals to the Boot-Unit when loading is completed
5. Either immediately or on some implementation specific cue, the Boot-Unit signals the CPU to begin normal operation. Prior to this step, the TAE is primed for use.



Normal operation

Once the CPU receives the ‘operate’ signal from the Boot-Unit, it cycles between the loaded BI3-TAE processes in a manner that ensures that each receives an equal share of instructions, as per abstraction criterion 3@A2a of TAIS.

Instructions executed by the BI3-TAE process may do the following:

- Change the state of the local registers in the CPU
- Change the state of the Random Access Memory for that BI3-TAE process only
- End the execution of the program. The CPU will pause on the halted program long enough so as not to speed up the

overall performance of the other BI3-TAE processes.

The actual specifics of the CPU architecture deliberately remain undefined at this point so as to maximise the deployment range.

Shutdown

Shutdown of the TAE can occur when the implementation triggers the `shutdown()` method of the `BI3TAE` class. At this point all BI3-TAE processes are halted immediately, the memory cleared and the system powered down.

Interfaces

```
public class BI3TAE2
{
    public void factoryLoad() ;
    public void boot() ;
    public void shutdown() ;
}
```

```
public class Process3
{
    public void start() ;
}
```

² Access to the BI3TAE class is limited to the physical hardware itself or the user account that launch the software Emulator

³ The BI3-TAE processes each have access to their own Process class