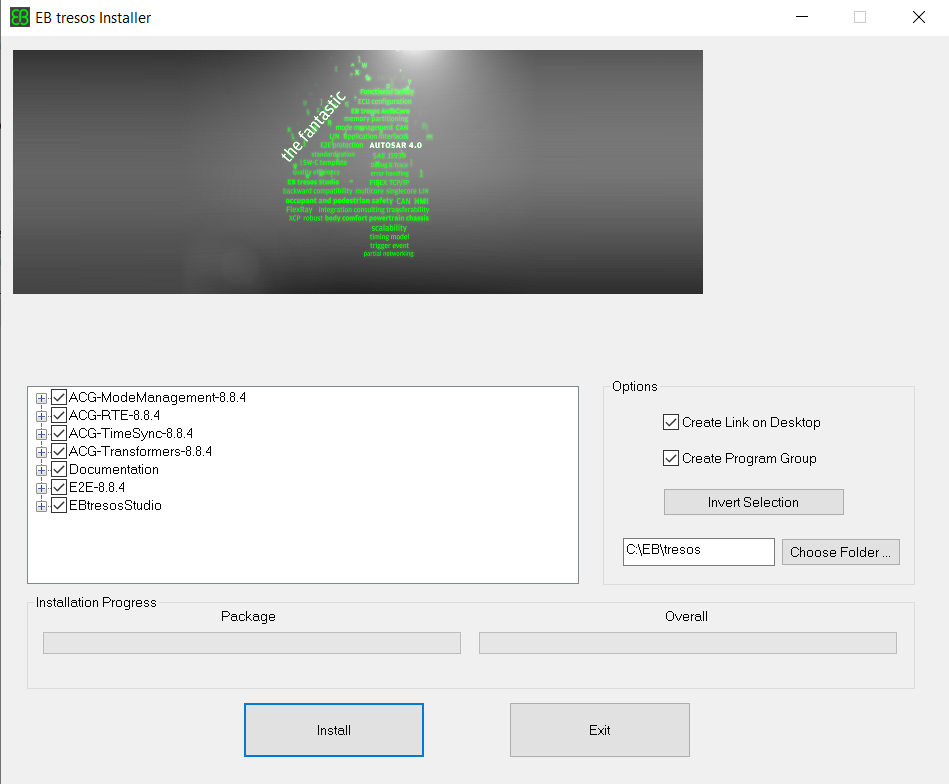
**Working with EB tresos Studio**

**Chương 1. Cài đặt phần mềm và tạo project**

**1. Cài đặt phần mềm**

B1. Truy cập vào website của EB tresos: <https://www.elektrobit.com/> download vào giải nén file tải về.

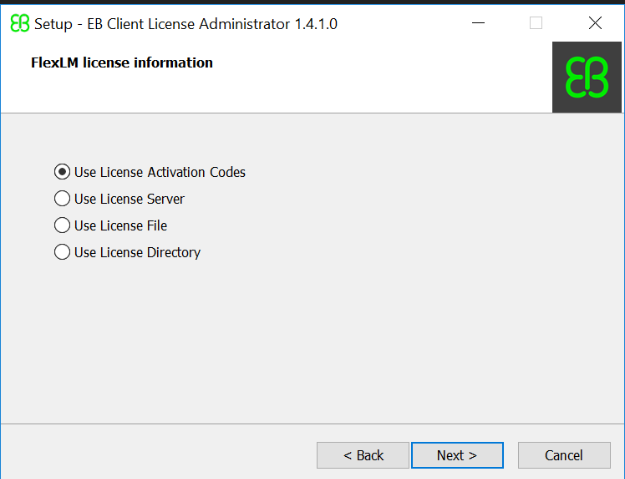
B2. Kích đúp vào file setup.exe và tiến hành cài đặt EB tresos



B3. Sau khi cài đặt thành công, tiến hành “add license” để có thể dùng được EB tresos.

B4. Cài đặt tool “EB Client License Administrator”

B5. Kích đúp vào file **EB\_Client\_License\_Administrator\_X\_Y\_Z\_Setup.exe** tiến hành cài đặt và add license. Ở đây chúng ta có nhiều sự lựa chọn về license (tùy thuộc vào loại license được cung cấp). Ở đây chúng ta đang sử dụng License Activation Codes.



B6. Add License

**\* License Activation Codes:**

- Kích đúp vào tool **EB Client License Administrator** và active code (ví dụ: 1239-E159-2D73-442F)

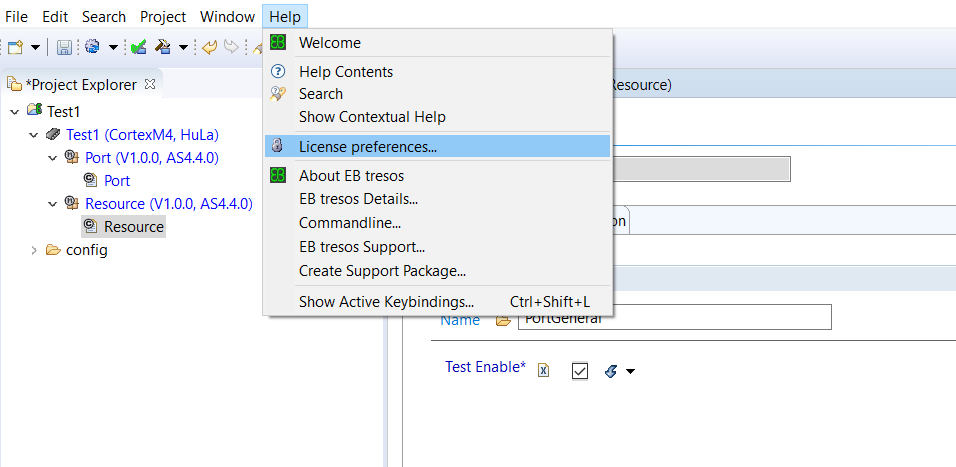


- Sau khi nhập code vào -> chọn **Create Active Request**

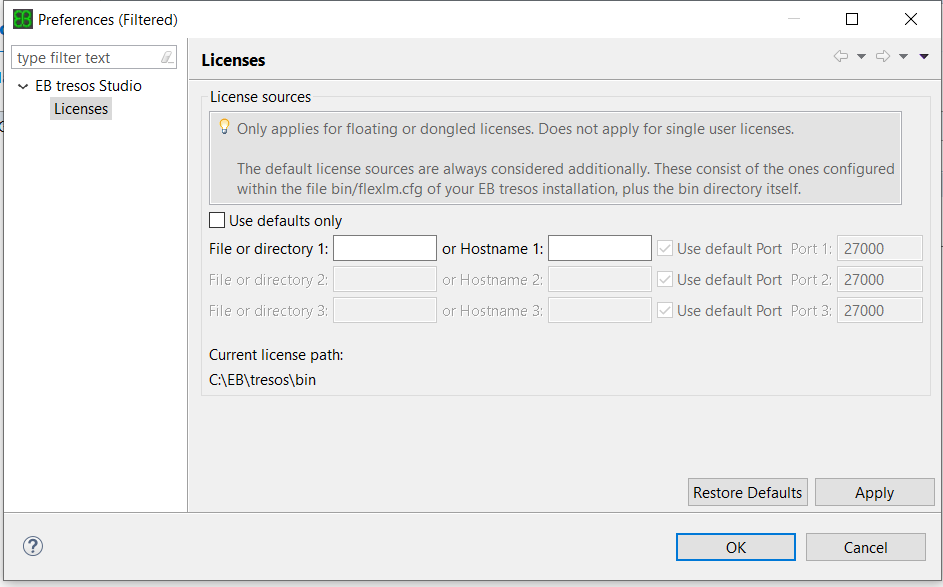
- Sau khi active code thành công, chúng ta có thể sử dụng được EB tresos.

**\* License Server**

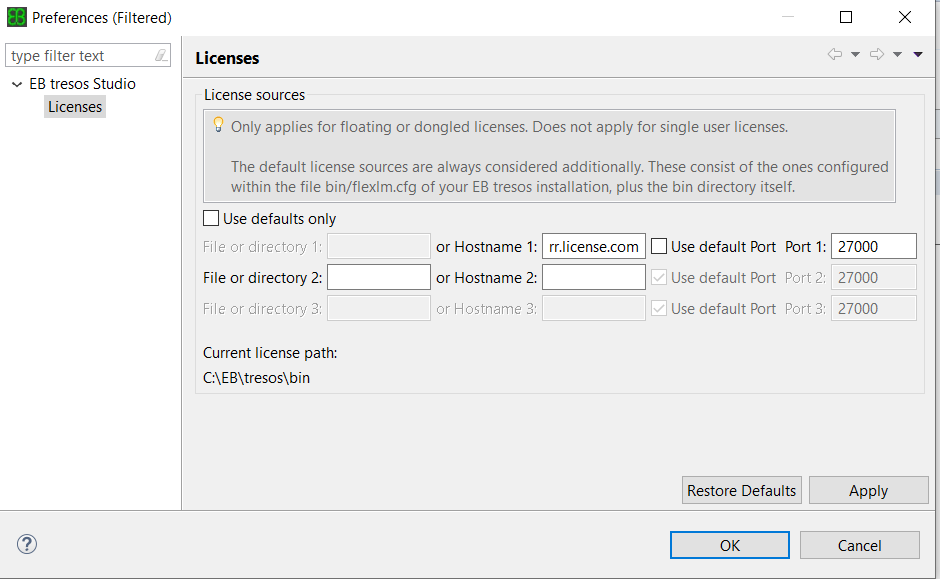
**-** Mở phần mềm EB tresos, trên thanh công cụ chọn **Hepl** -> **License Refernce**



- Sau đó xuất hiện một hộp thoại và bỏ chọn **Use default only**



- Ở đây chúng ta chúng ta sẽ điền vào ô **Hostname1** licsense mà được cấp và bỏ chọn **Use default Port** (được cấp)

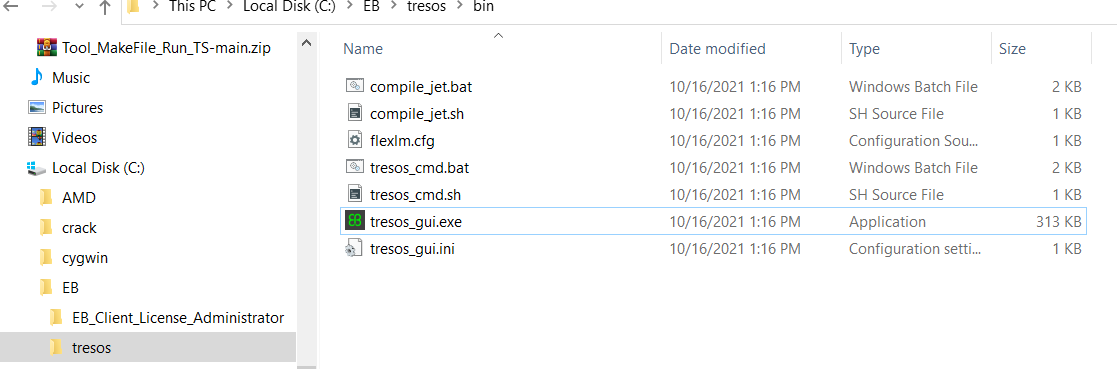


- Nhấn chọn **Apply** để active thành công sau đó chọn **OK**.

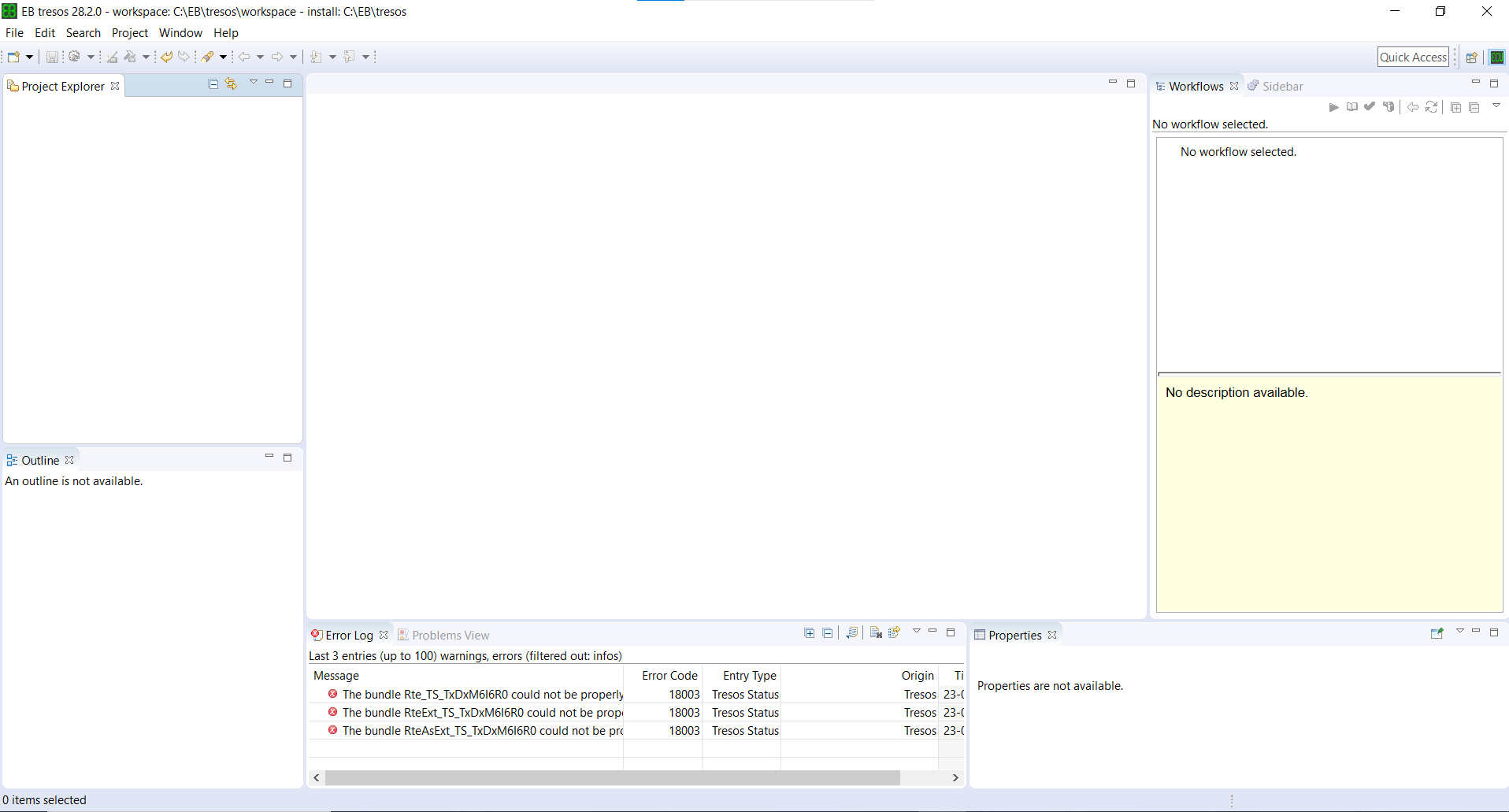
\* **Lưu ý**: *EB tresos là phần mềm có bản quyền, vì vậy chúng ta cần phải đăng kí License (được sử dụng free trong 3 tháng từ ngày được cấp License), nếu chúng ta tiếp tục sử dụng thì cần phải tốn phí. Đăng kí License trên website* <https://www.elektrobit.com/>

**2. Tạo project với EB tresos**

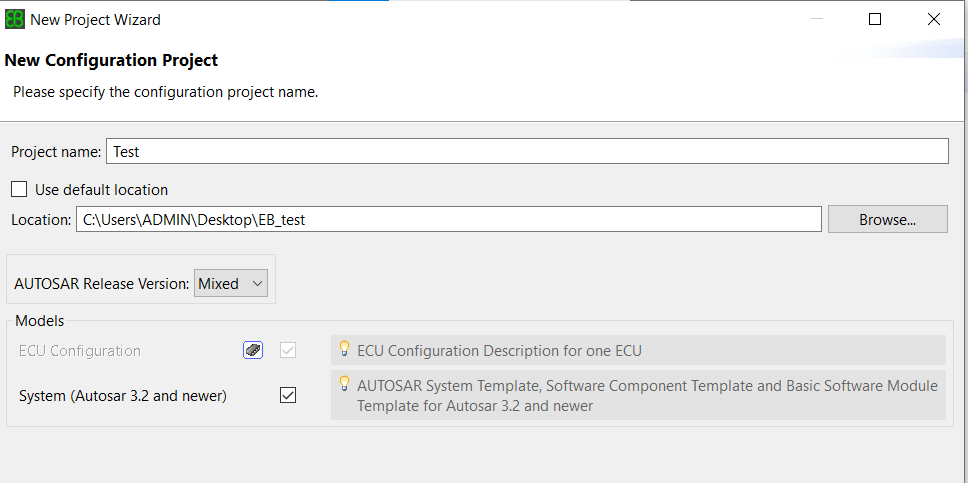
B1. Mở phần mềm EB tresos trên màn hình desktop hoặc kích đúp vào file tresos\_gui.exe



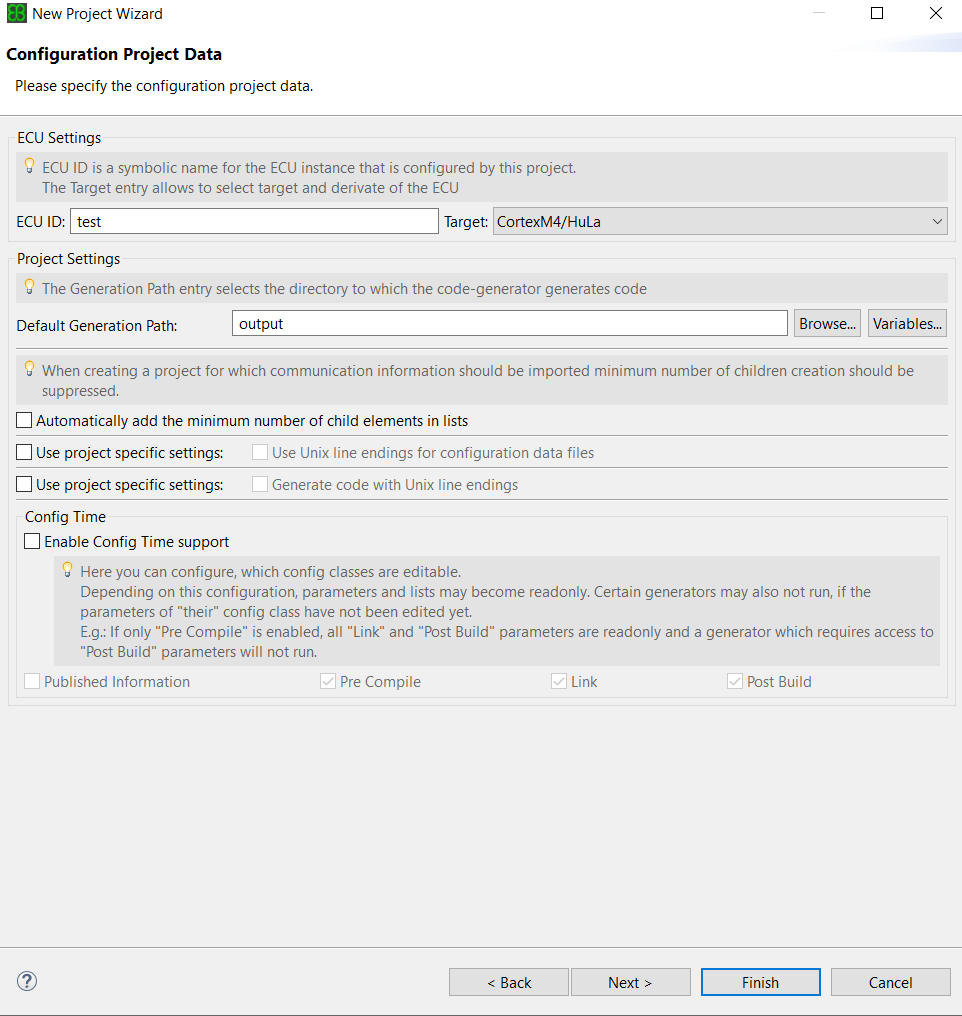
B2. Xuất hiện giao diện EB treso và tiến hành tạo project



B3. Vào **File** -> **New -> Configuration Project** xuất hiện hộp thoại như hình, đặt tên project và lưu project



B4. Sau đó chọn **Next**, và thiết lập **ECU,** điền ECU ID (đặt tên bất kì) và chọn **Target** (Chip được sử dụng trong project) sau đó chọn **Next**.

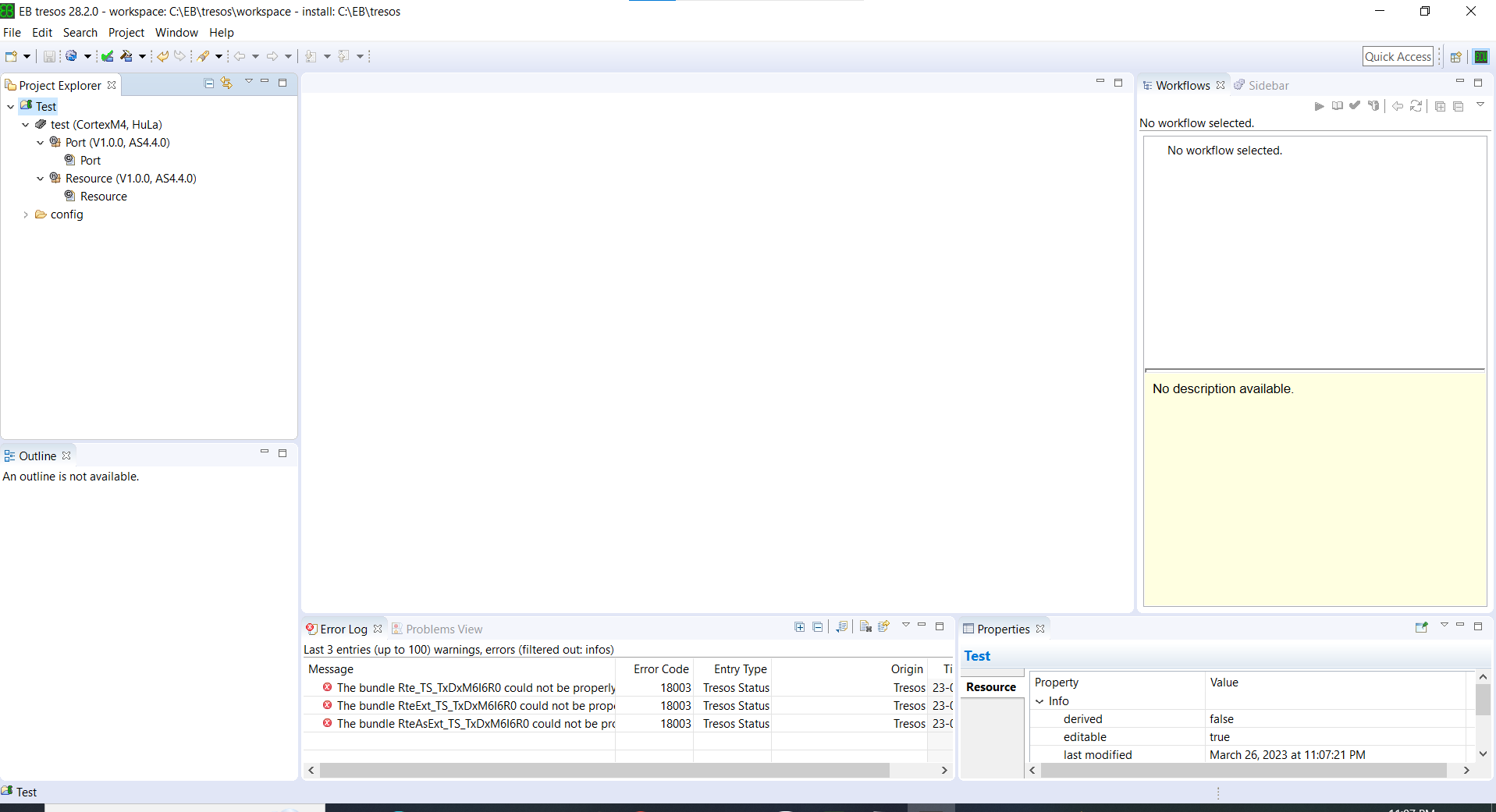


B5. Chọn Module bạn muốn sử dụng trong project



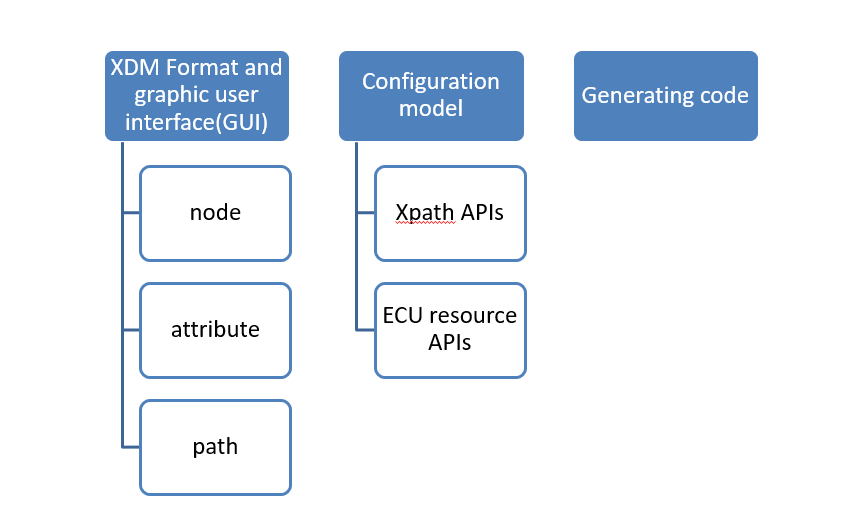
* **Available Modules:** Những module có sẵn trong **plugins**
* **Module configuration:** Module được sử dụng để config cho project
* **Details:** Mô tả về module

B6. Sau đó chọn **Finish** => tạo thành công project.



**Chương 2. The XDM format**

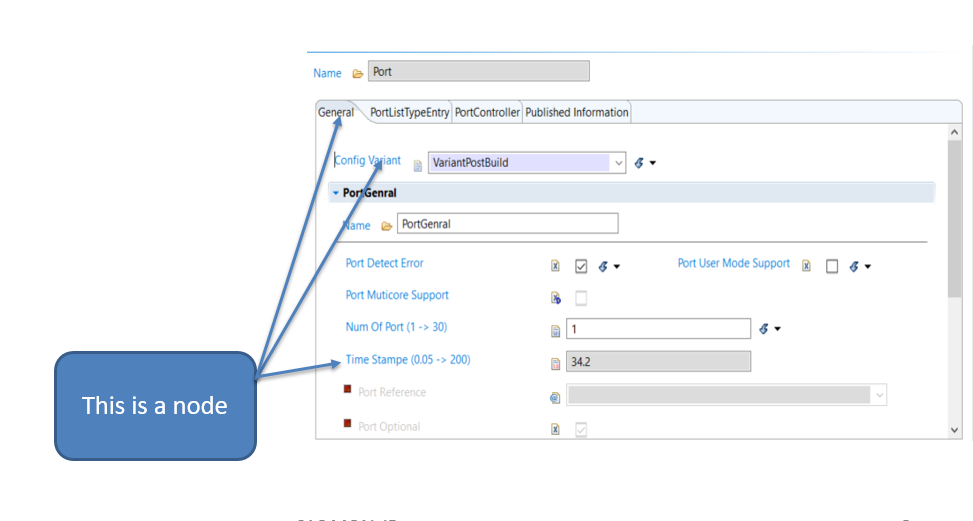
**2.1. Tổng quan về Format XDM**

****

Trong một file XDM gồm các thành phần chính: node (schema node, data node), attribute và path. DataModel hỗ trợ các loại node khác nhau:

* schema node: sẽ định các tham số cấu hình và cấu trúc của các tham số cấu hình. Xác định type, structure and attribute mặc định cho data-nodes.
* data node: Các nút dữ liệu đại diện cho các tham số cấu hình.

**2.1. Node**

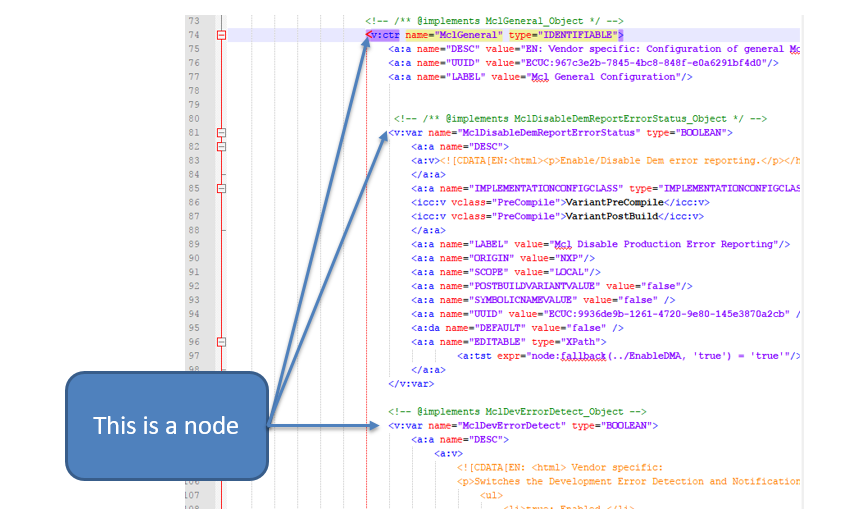
****

Hình 1. Các node trên EB tresos

Các **node tag** dùng để tạo ra các node trong datamodel. Các nhiều loại node tag như là (container, variable, list, ...).

Các thuộc tính XML cho một node bao gồm:

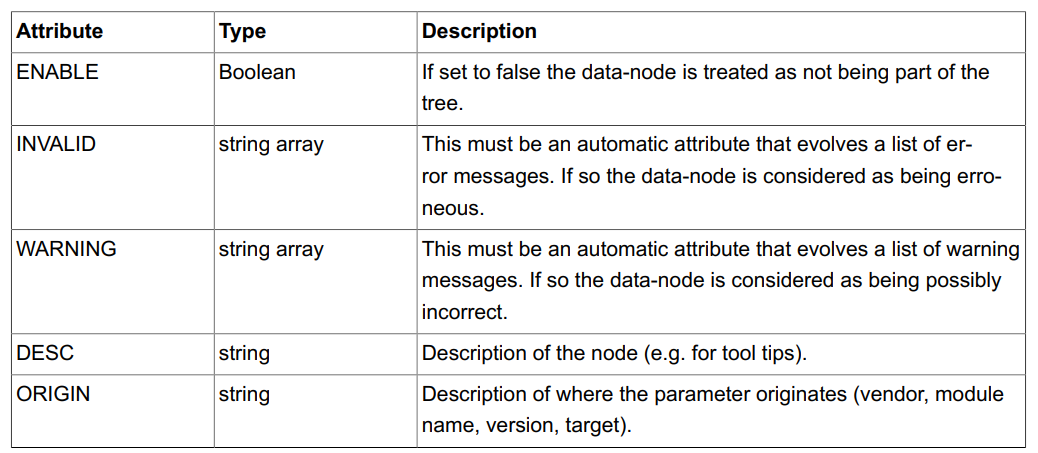
* name: Tên của một **node**
* type: thuộc tính được thêm vào để cho **type** node
* value: Mỗi nút có thể có một giá trị được đặt với thuộc tính.
* factory



Hình 2. Các node trong file xdm

Một node tag sẽ bắt đầu với **v:tag** (v:list, v:ctr, v:var, …)

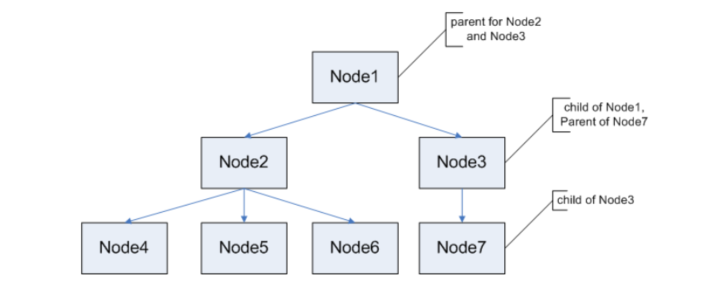
Các data attribute có thể sử dụng cho các schema node:



Loại node được xác định bởi **name tag** (crt, chc, lst, var...). Bất kỳ node nào cũng có thể có phần tử con được gọi là **node** **child.** Các node child có thể là bất kỳ node nào với các loại khác nhau. Hình bên dưới thể hiện các loại node thường được sử dụng trong file XDM.

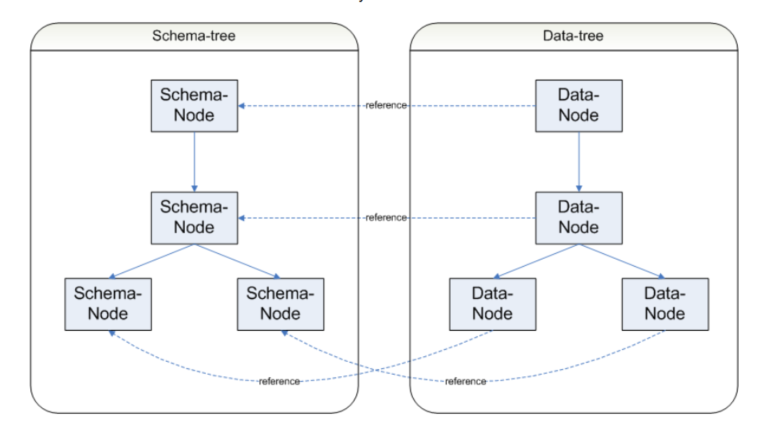
Hình 3. Các loại node trong EB tresos

Tất cả các nút được lưu trữ trong một **tree structure**. Vì vậy, chỉ có một node **parent** và nhiều node **child.**



Hình 4. Cấu trúc của các node trong EB tresos

Mỗi data-node sẽ có một một schema-node mô tả nó. Mỗi data-node sẽ tham chiếu đến sechema node mô tả nó.



Hình 5. Cấu trúc của các node trong EB tresos

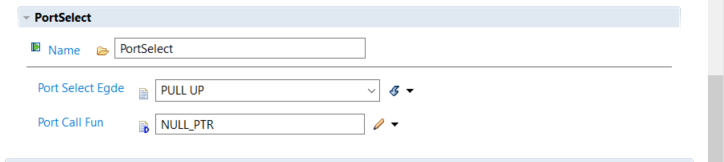
**2.1.1. Node container (ctr)**

- Stand for “container”

- Tất cả code trong tệp xdm đều có tag <v:ctr ... > và kết thúc </v:ctr> là một vùng chứa.



Hình 6. Node container trong file xdm



Hình 7. Node container hiển thị trên EB tresos

- Các **tag child** có thể bên trong tag ctr là: v:ctr, v:chc, v:list, v:var, v:ref.

**2.1.2. Node choice (chc)**

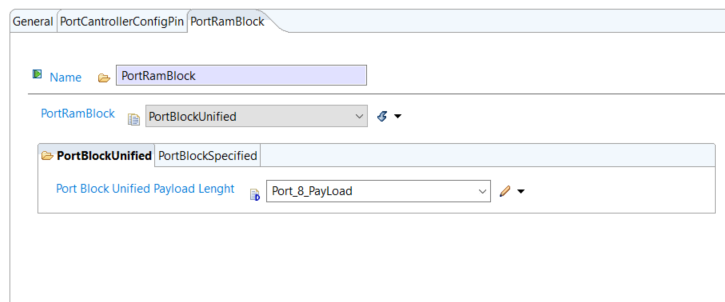
- v:chc được định nghĩa một schema node được chọn. v:chc sẽ chứa v:ctr node.

- Bên trong cấu hình chỉ có thể thêm một container data node vào một lựa chọn phải phù hợp với một trong các vùng chứa của choice schema-node.

- Tag child bên trong v:chc chỉ có thể là v:ctr



Hình 8. Node choice trong file xdm



Hình 9. Node choice trên EB tresos

Qua hình trên ta thấy chỉ một trong hai node “<v:ctr name="PortBlockUnified" …” hoặc “PortBlockSpecified” xuất hiện.

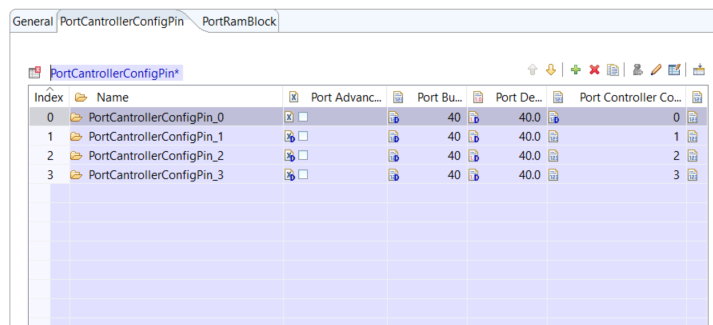
**2.1.3. Node list (lst)**

- Stand for “list”

- Tất cả code trong tệp xdm đều có tag <v:lst ... > và kết thúc </v:lst> là một vùng chứa.

- Một danh sách trong schema chỉ có một **node child** có thể được sao chép trong

dữ liệu nhiều lần. List schema-nodes có thể sử dụng để tạo các tables.



Hình 10. Node list trên EB tresos

- List có hai variant: list và map sẽ chọn type cho list schema node.

\* Với type=” ” : tạo ra một node list, một list chỉ chứa một danh sách các mục được sắp xếp theo thứ tự. Với type này luôn luôn có một cột bên cạnh cột index. Với list mà type=’’ ’’: các node child hợp lệ là: v:var và v:ref.

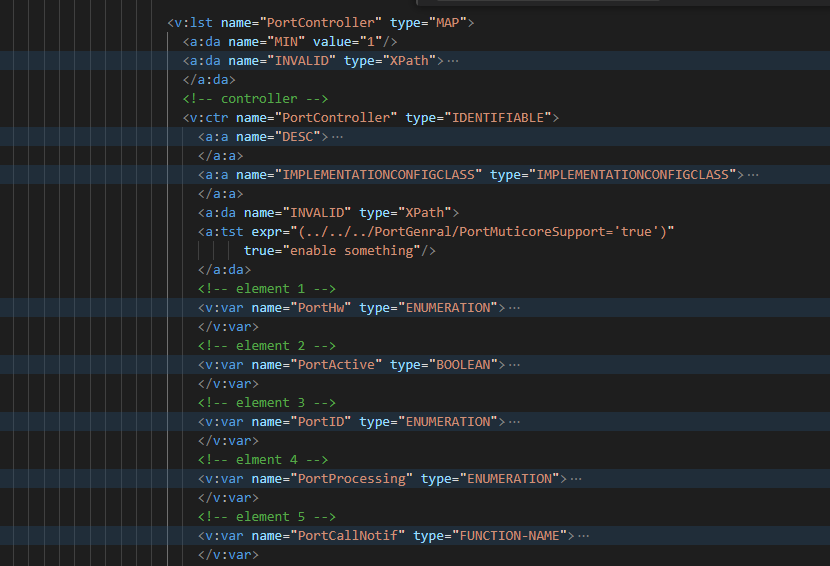


Hình 11. Node list với type=’’ trong file xdm

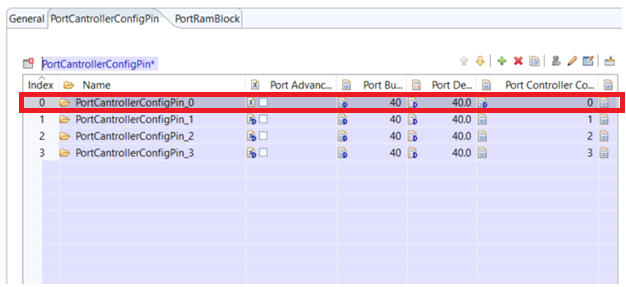


Hình 12. Node list với type=’’ trên EB tresos

\* Với type=’’MAP’’: tạo ra một map. Map là một danh sách có thứ tự trong đó tên của tất cả các mục là duy nhất trong danh sách. Một map có thể chứa một số cột nơi có thể định cấu hình mức độ hiển thị của các cột mặc định.



Hình 13. Node list với type=’MAP’ trong file xdm



Hình 14. Node list với type=’MAP’ trên EB tresos

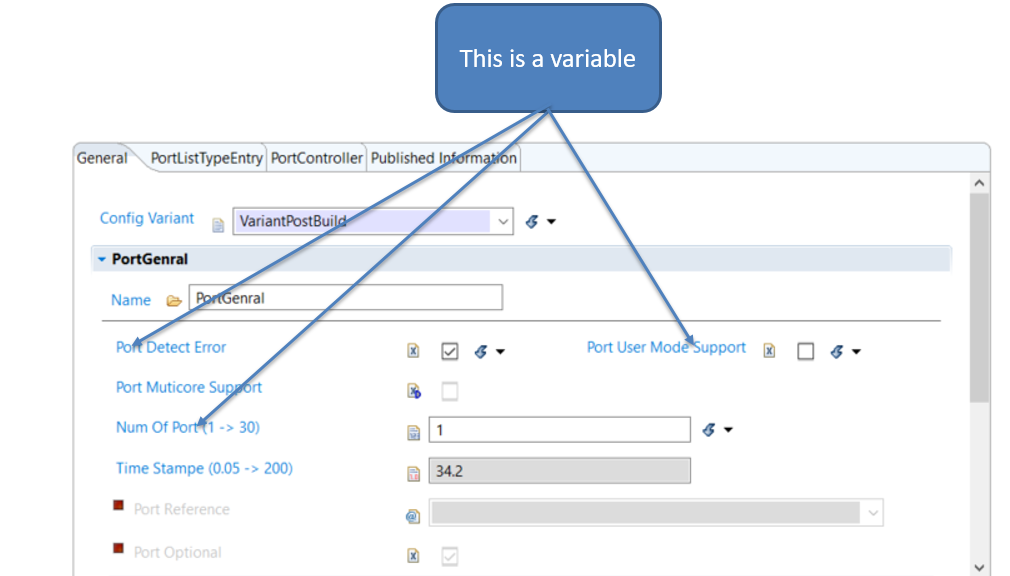
-Với list mà type=’’MAP’’ các node child hợp lệ là: v:ctr và v:chc

**2.1.4. Node varibale (var)**

- Stand for “var”

- Tất cả code trong tệp xdm đều có tag <v:var... > và kết thúc </v:var> là một vùng chứa.

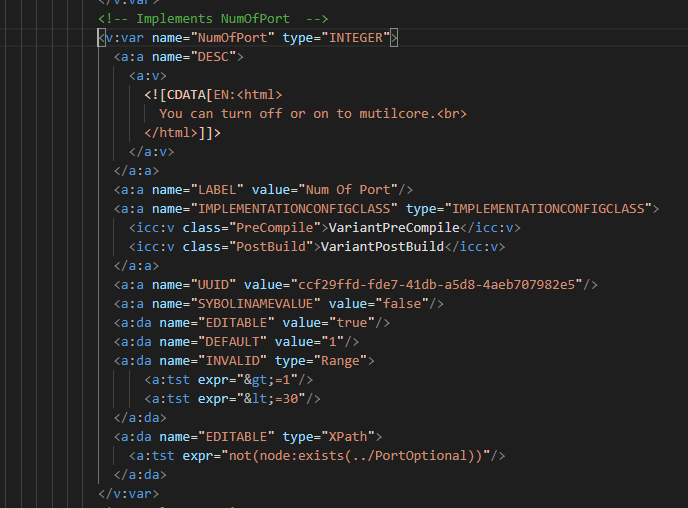
-Tạo một variable schema node. variable schema node xác định tham số cấu hình có thể lưu trữ một giá trị.



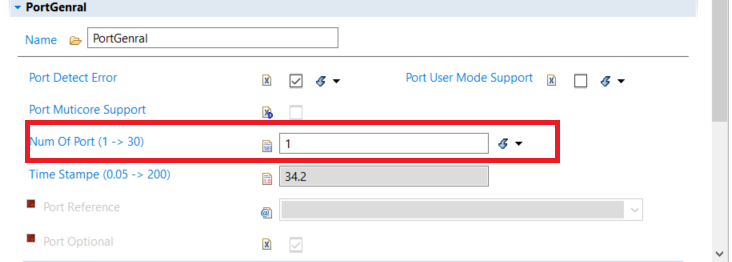
Hình 15. Node var trên EB tresos

- **Type** của **value** được xác định bởi loại thuộc tính XML của tag v:var. Các loại sau đây là được hỗ trợ:

+ INTERGER: Có thể lưu trữ một số nguyên có 64 bit. Giá trị số nguyên được mã hóa ở định dạng thập phân.

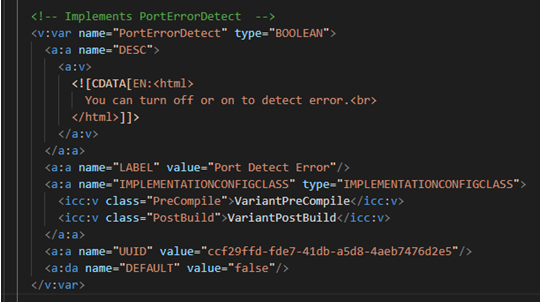


Hình 16. Node var với type=” INTERGER” trong file xdm

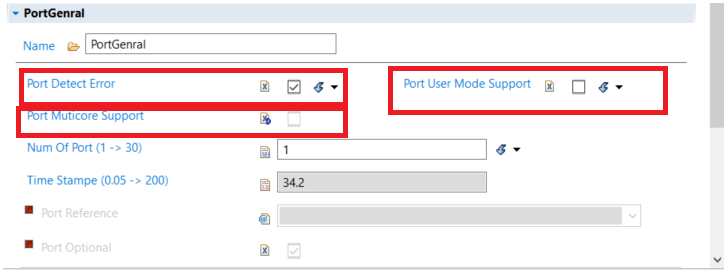


Hình 17. Node var với type=” INTERGER” trên EB tresos

+ BOOLEAN: sẽ lưu trữ giá trị BOOLEAN. Giá trị mã hóa của true hoặc false được xác định bởi thuộc tính RANGE. Giá trị đầu tiên của thuộc tính RANGE được định nghĩa là giá trị true và giá trị thứ hai là false.

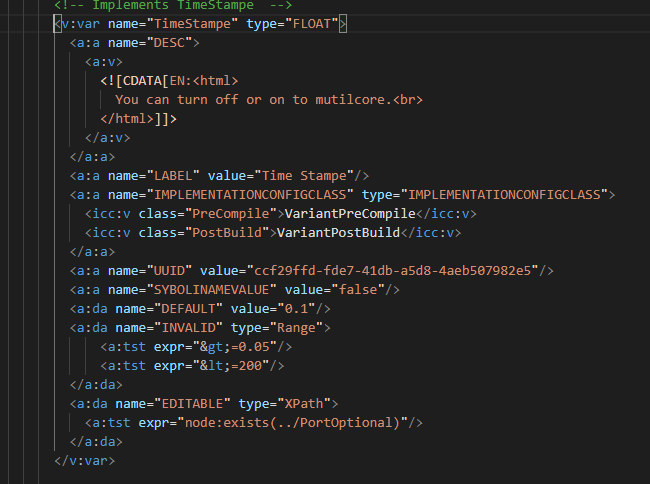


Hình 18. Node var với type=” BOOLEAN” trong file xdm

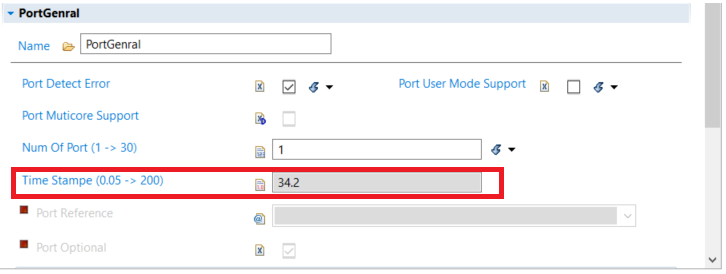


Hình 19. Node var với type=” BOOLEAN” trên EB tresos

+FLOAT: lưu trữ giá trị float có 64 bit. Các giá trị float được mã hóa ở định dạng thập phân ([ "-" ] <chữ số> "." <chữ số>).

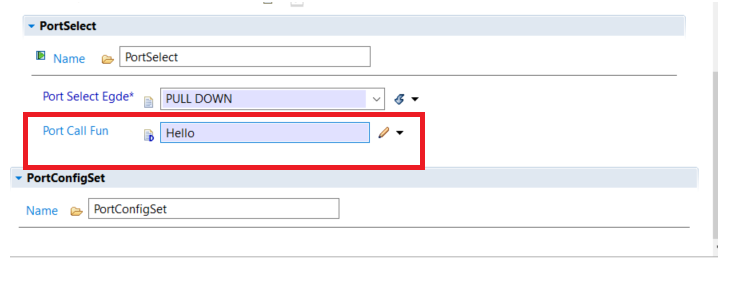


Hình 20. Node var với type=” FLOAT” trong file xdm



Hình 21. Node var với type=” FLOAT” trên EB tresos

+ STRING: Lưu trữ một chuỗi ASCII tùy ý.



Hình 22. Node var với type=” STRING” trên EB tresos

+ MULTILINE-STRING: Lưu trữ một chuỗi ASCII tùy ý vượt quá một dòng.

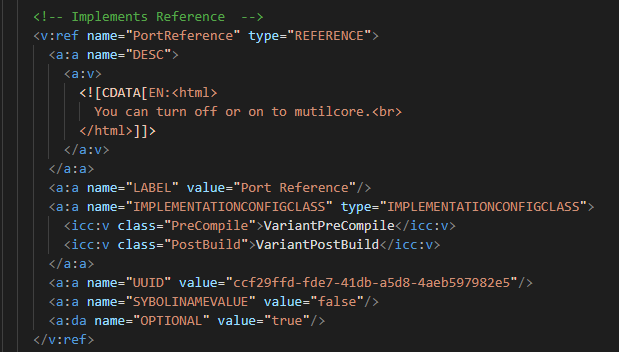
- Với node var thì không có **node child.**

**2.1.4. Node reference (ref)**

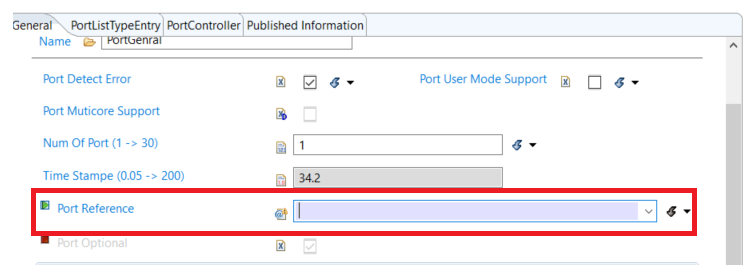
- Stand for “ref”

- Tất cả code trong tệp xdm đều có tag <v:ref... > và kết thúc </v:ref> là một vùng chứa.

- Tạo ra một reference schema-node. Reference schema-node sẽ định nghĩa một tham số cấu hình mà refer tới thông số cấu hình khác.



Hình 23. Node ref trong file xdm



Hình 24. Node ref trên EB tresos

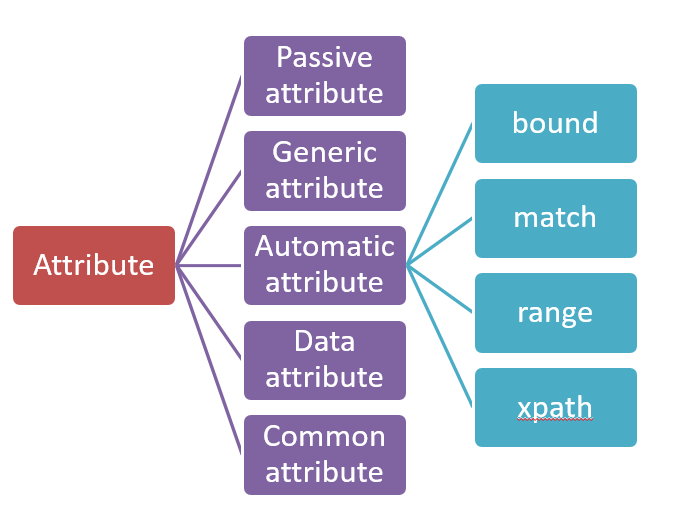
**2.2. Attribute**

- Attribute của xdm không phải là xml attribute

- Attribute có tên và có thể có nhiều giá trị chuỗi.

- Tất cả các tag bắt đầu bằng <a:da> hoặc <a:a> là thuộc tính của node.

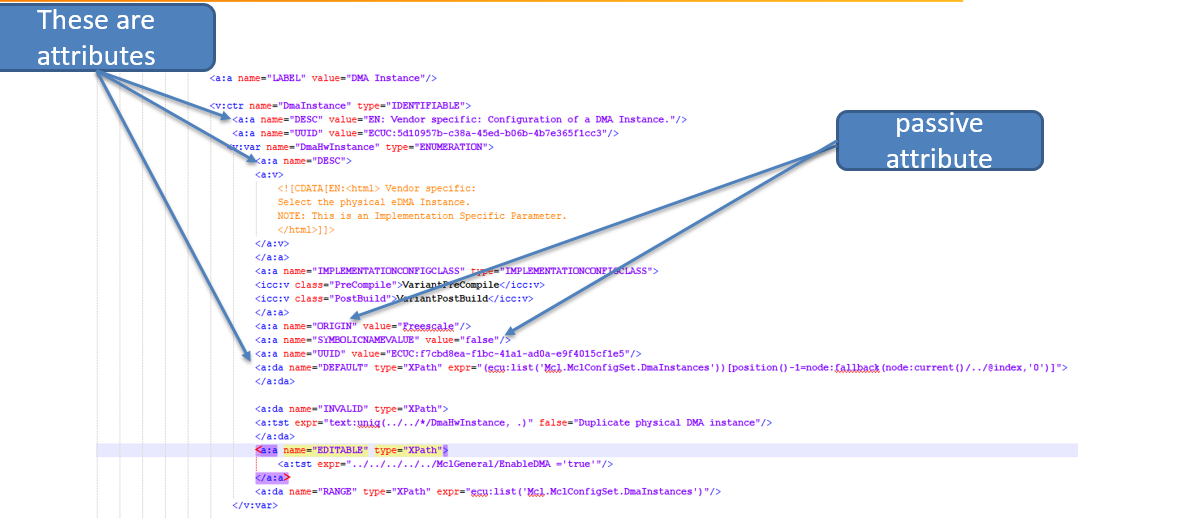
- Attribute có ba dạng: Passive attributes, Generic attributes, Automatic attributes.



Hình 25. Các atributes trong EB tresos

**2.2.1. Passive attributes**

Tham khảo tài liệu “2.4\_Studio\_documentation\_developers\_guide.pdf”

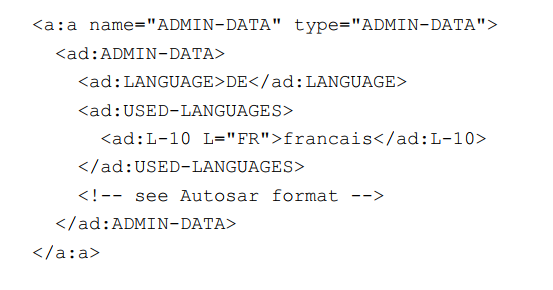


Hình 26. Passive atributes

**2.2.2. Generic attributes**

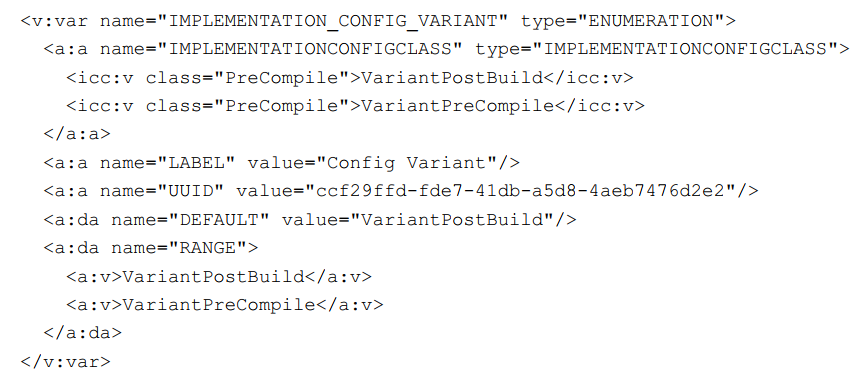
Generic attributes bao gồm:

- **ADMIN-DATA** :Thuộc tính chung "ADMIN-DATA" được sử dụng đặc biệt để giữ cấu trúc dữ liệu phức tạp của thuộc tính đó được định nghĩa bởi Autosar một cách chung chung.

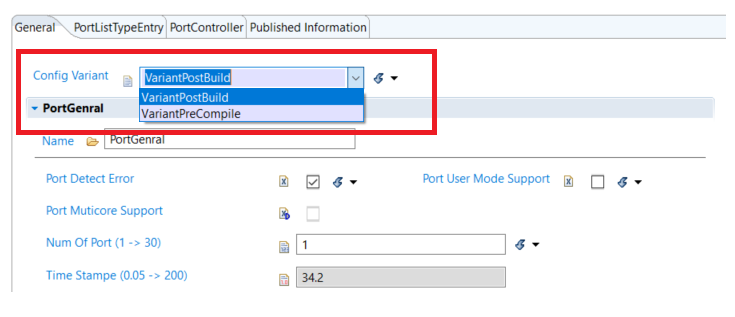


Hình 27. Generic atributes thuộc tính "ADMIN-DATA"

- **Implementation Configuration Variant**: Mỗi "MODULE-DEF" có thể chứa tham số "IMPLEMENTATION\_CONFIG\_VARIANT" xác định các variant cấu hình được hỗ trợ của mô-đun thông qua RANGE của nó.

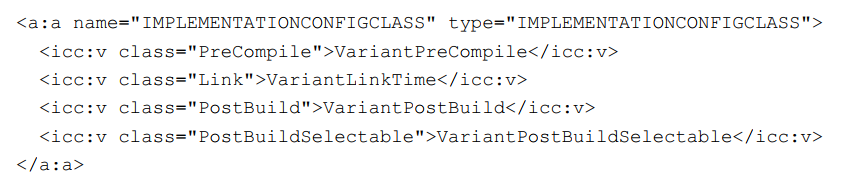


Hình 28. Trong file xdm



Hình 29. Trên EB tresos

- Implementation Config Class: Thuộc tính IMPLEMENTATIONCONFIGCLASS giữ lớp và variant làm giá trị.



**2.2.3. Auto attributes**

Automatic attributes được chia làm 7 loại khác nhau: Bound, Match, Range, Xpath, Multi test, EcucCond, EcucValidCond. Tuy nhiên có hai loại quan trọng và thường dùng nhất là Range và Xpath. Để hiểu rõ hơn về các loại còn lại có thể tham khảo tài liệu (2.4\_Studio\_documentation\_developers\_guide.pdf). Ở đây chúng ta sẽ đi tìm hiểu về hai loại thường dùng nhất.

1. *Range Attributes*

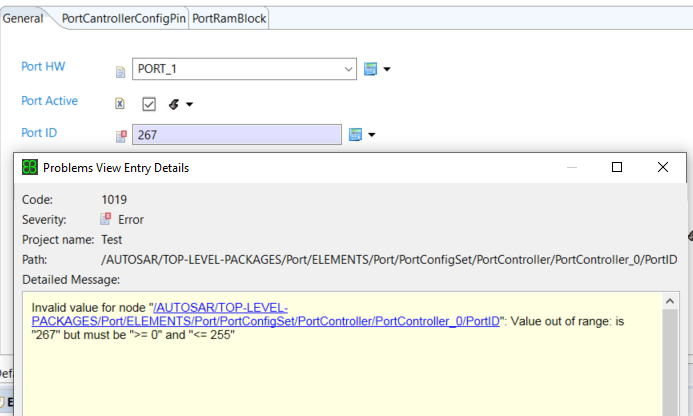
Range Attributes xác định giới hạn để kiểm tra một giá trị có hợp lệ hay không. Nó xác định phạm vi mà giá trị của node có thuộc tính Range có thể có. Nếu mà giá gị của node không nằm trong phạm vi hợp lệ sẽ báo lỗi.

Dưới đây là môt ví dụ về Range Attribute:



Hình 30. Range Attribute trong file xdm

Khi chúng ta nhập một giá trị vượt qua mức yêu cầu thì trên EB tresos sẽ report lên lỗi như hình bên dưới.



Hình 31. Report ra lỗi khi nhập quá giới hạn

**Chú ý:** *Đừng nhầm lẫn các Range Attributes của loại "Range" với thuộc tính "RANGE" hạn chế các giá trị hợp lệ của node của nó nhé.*

1. *Xpath attribute*

Xpath không chỉ là XPath là đường dẫn XML. Nó là một cú pháp để tìm kiếm bất kỳ element nào trong ebtresos bằng cách sử dụng biểu thức XML path. XPath được sử dụng để tìm locator của bất kỳ element nào bằng cách sử dụng các Xpath API.

**\* Xpath attribute (DEFAULT)**

**DEFAULT** attribute có nghĩa là tạo ra giá trị mặc định cho Node. Xpath ở đây có nghĩa là giá trị mặc định của Node sẽ được tính toán bởi biểu thức sau expr.



Hình 32. Xpath attribute (DEFAULT)

Giá trị mặc định của Node “PortID” sẽ được lấy sau biểu thức expr.

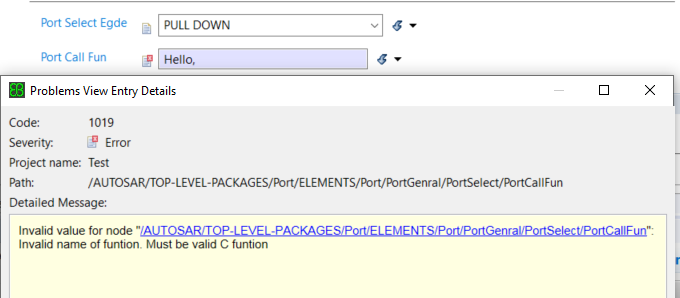
**\* Xpath attribute (INVALID)**

**INVALID** attribute có nghĩa là giá trị của một node có hợp lệ hay không. Xpath có nghĩa là giá trị của node hiện tại sẽ được kiểm tra bởi biểu thức sau expr. Nếu mà kết quả trả về là false thì sẽ xuất hiện lỗi trên EB tresos.

****

Hình 33. Xpath attribute (**INVALID**)

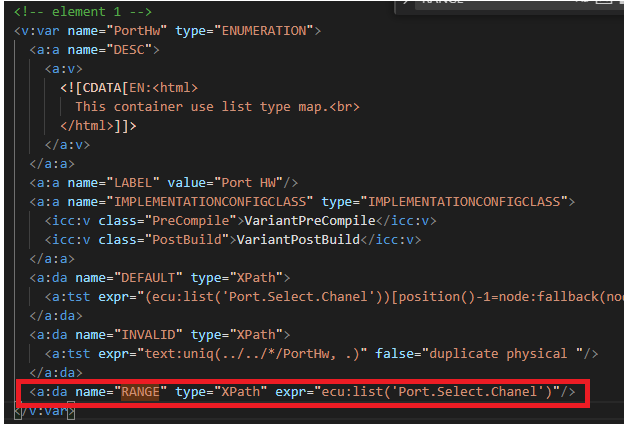
Khi chúng ta nhập giá trị cho Node nếu giá trị không thoải mãn các điều kiện sau biểu thức expr thì sẽ report lên lỗi. Ví du khi nhập dư dấu phẩy thì sẽ report lên lỗi như hình dưới.



Hình 34. Xpath attribute (**INVALID**)

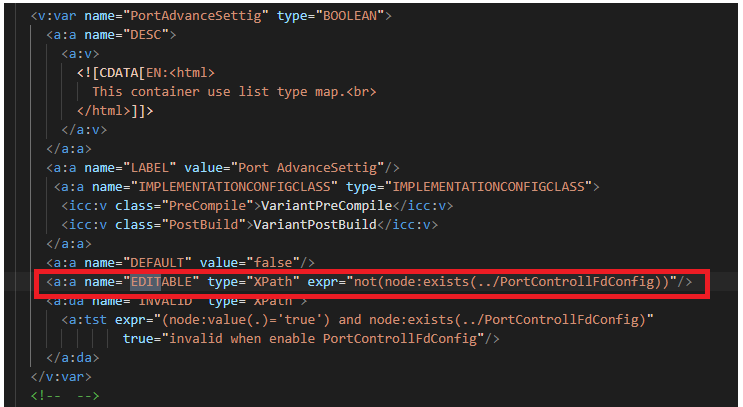
**\* Xpath attribute (RANGE và EDITABLE)**

* **RANGE** **atrributes**: giá trị của node hiện tại chỉ cho phép nằm trong giới hạn được xác định.



Hình 35. Xpath attribute (**RANGE**)

* **EDITABLE** attributes: có nghĩa là node hiện tại chỉ được chỉnh sửa khi giá trị của node hợp lệ.



Hình 36. Xpath attribute (**EDITABLE**)

**\* Xpath attribute (VISIBLE)**

Nếu giá trị sau biểu thức sau expr trả về true thì node đó sẽ bị ẩn đi (Không xuất

hiện trên EB tresos).

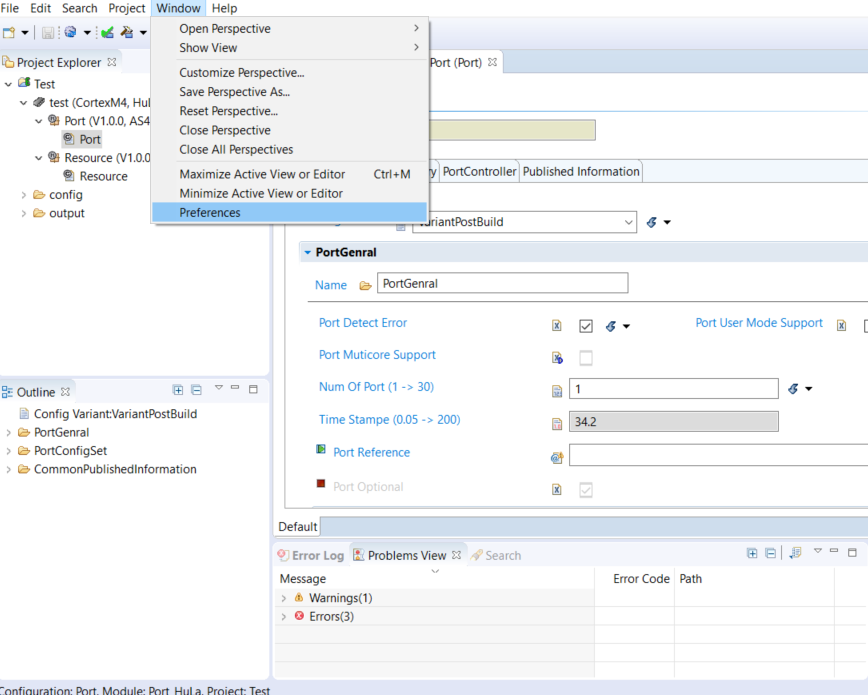


Hình 37. Xpath attribute (**VISIBLE**)

**2.2.4. Các XPath API thường được dùng trong EB tresos**

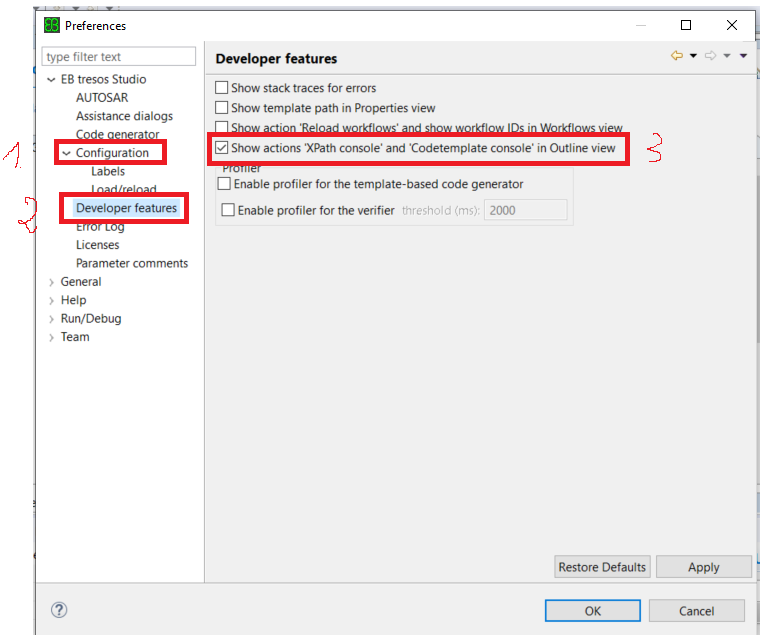
Đầu tiên, mình sẽ giới thiệu về **XPath Console** trên EB tresos. **XPath Console** đơn giản nó như là một cung cụ để **Debug** cho XPath API, giúp chúng ta kiểm tra việc xem giá trị của một node hay vị trí của node,… Để mở được hộp thoại chúng ta thực hiện các bước sau:

B1. Chọn Windows trên thanh công cụ -> **Reference**

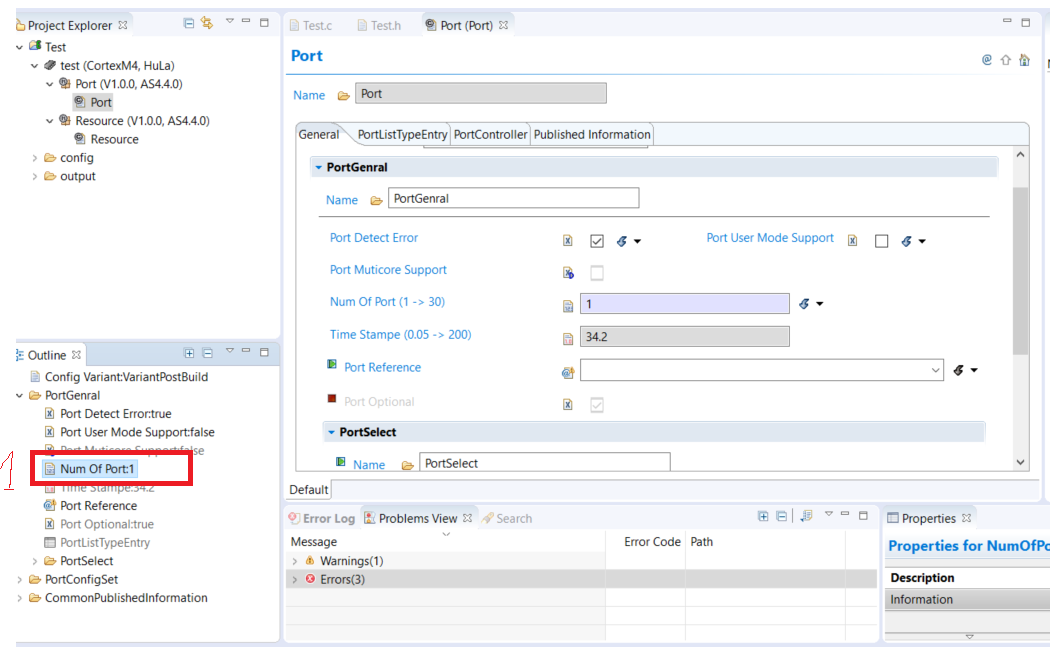


Hình 38. Show XPath Console

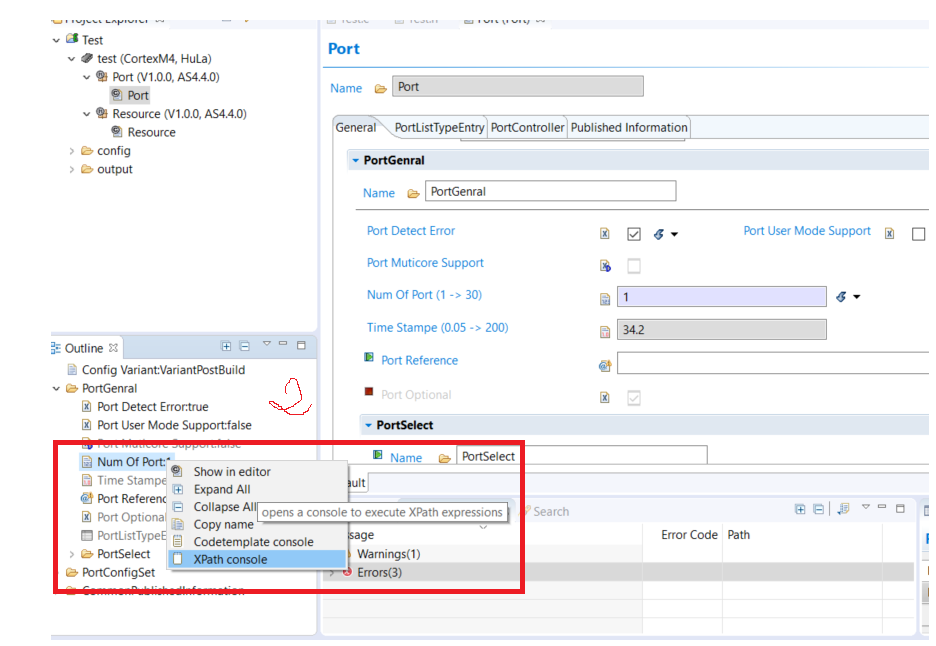
B2. Xuất hiện một hộp thoại chọn **Configtion -> Develop Features -> tick vào ô thứ 4.** Sau đó chọn **OK.**

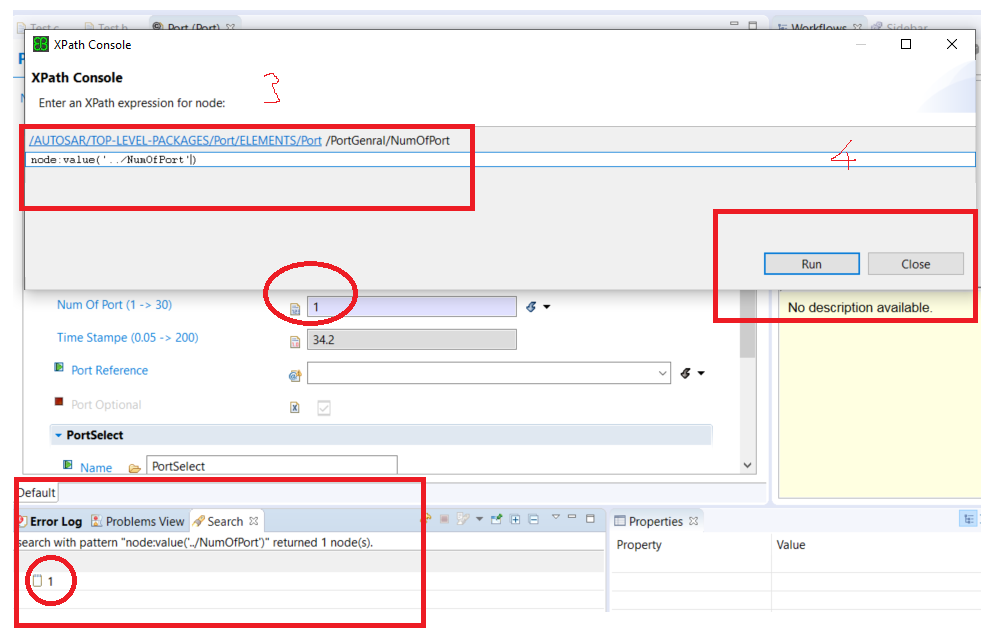


B3. Ví dụ để xem giá trị của một node chúng ta làm như sau:



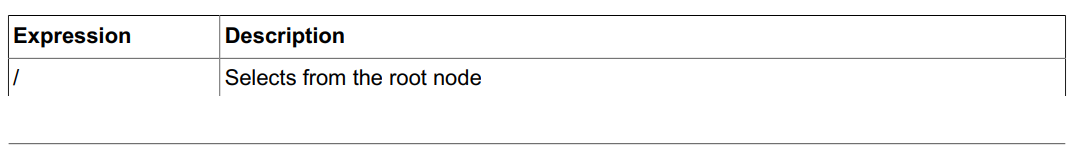
Kích chuột phải chọn XPath Console

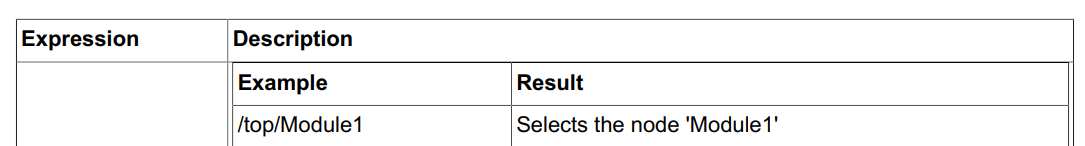


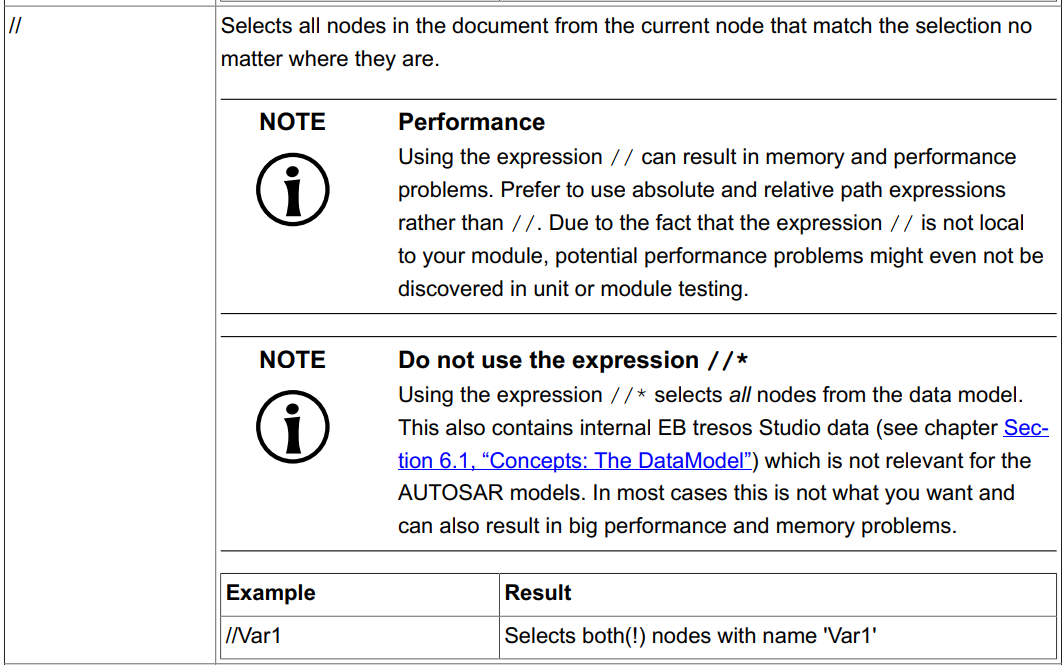


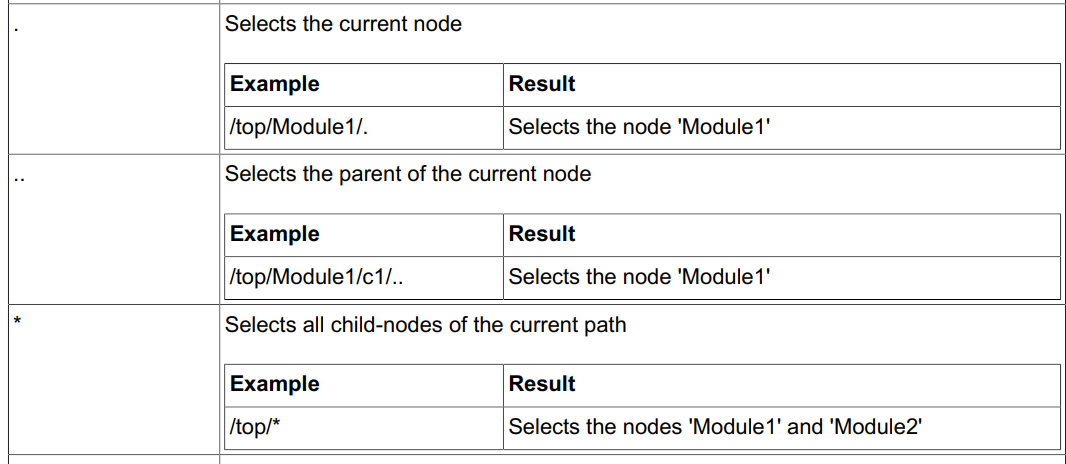
*a. XPath expressions*

Để chọn một Node, có thể sử dụng các đường dẫn tuyệt đối hoặc tương đối với các biểu thức sau:



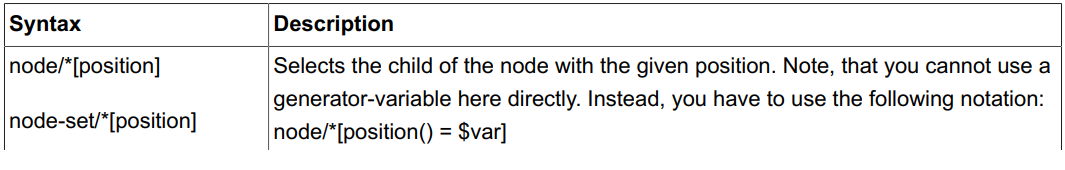


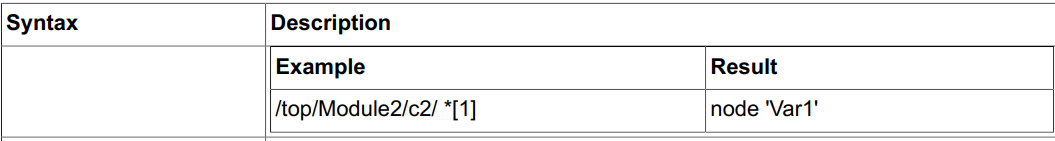




*b. XPath predicates*

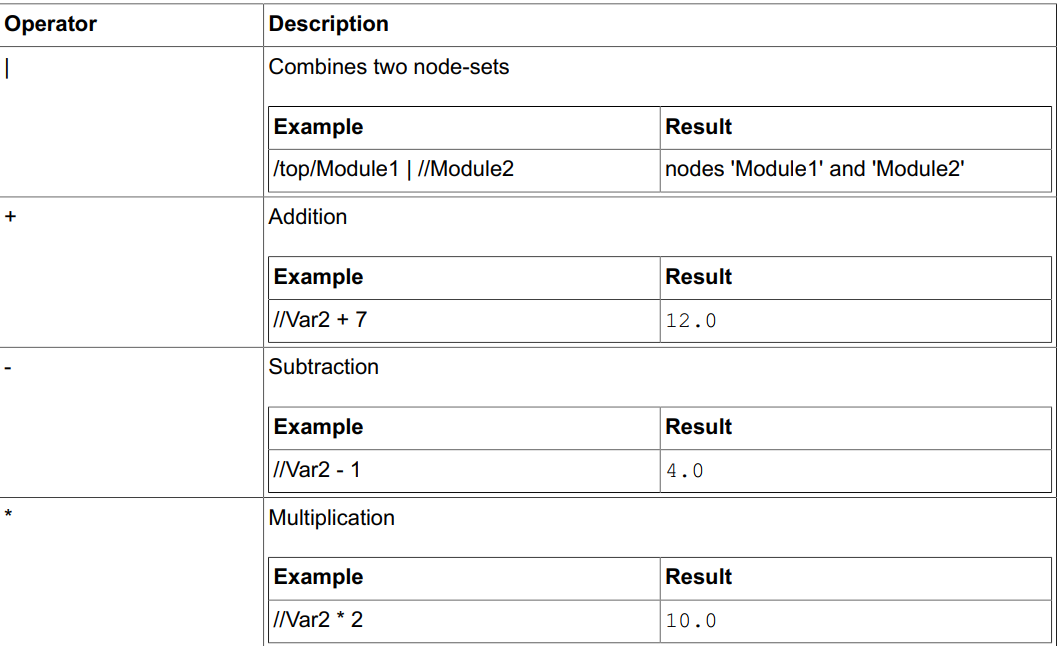
Các vị từ được sử dụng để tìm một Node cụ thể hoặc một Node chứa một giá trị cụ thể. Vị ngữ luôn luôn trong ngoặc vuông.

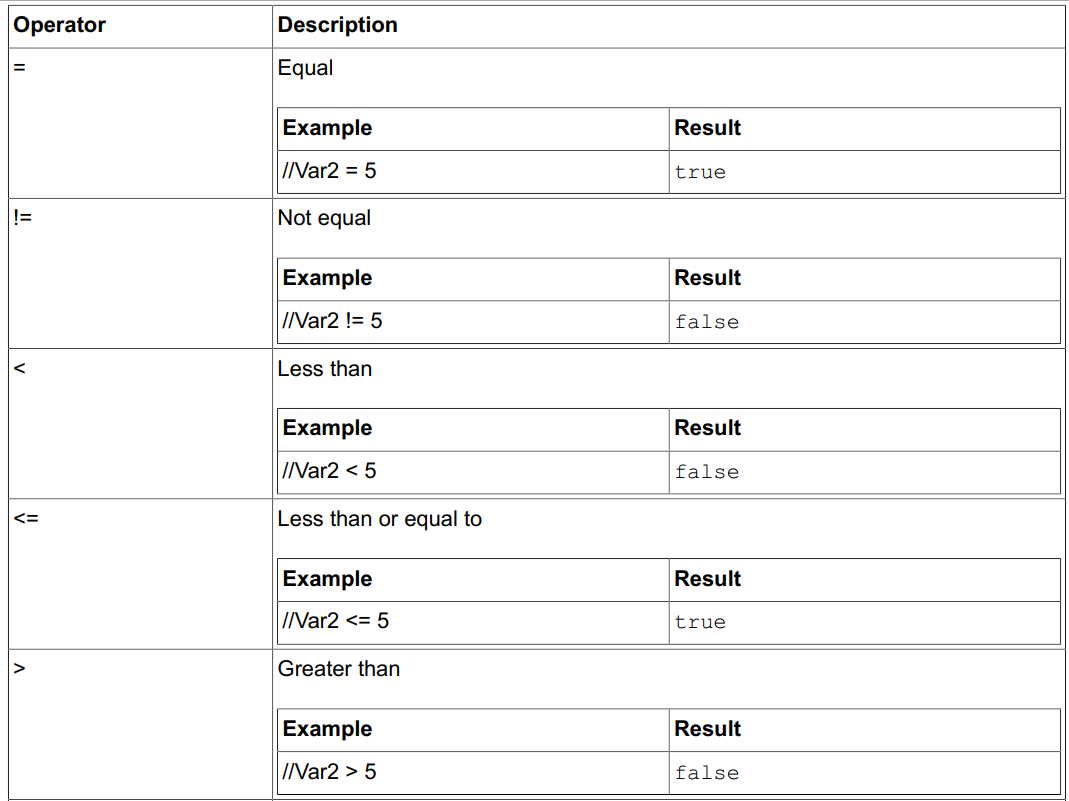


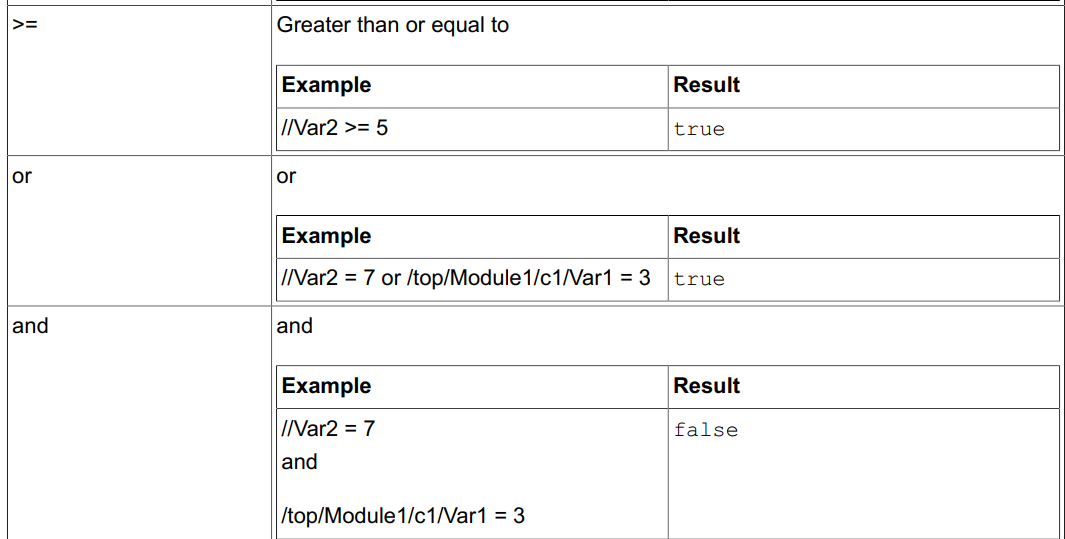


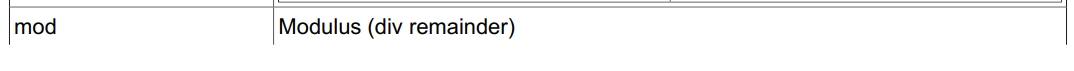
*c. XPath operations*

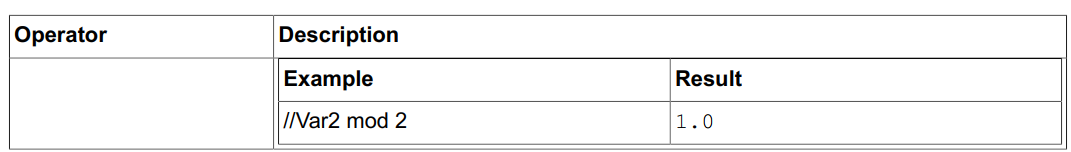
Dưới đây là danh sách các toán tử có thể được sử dụng trong biểu thức XPath:



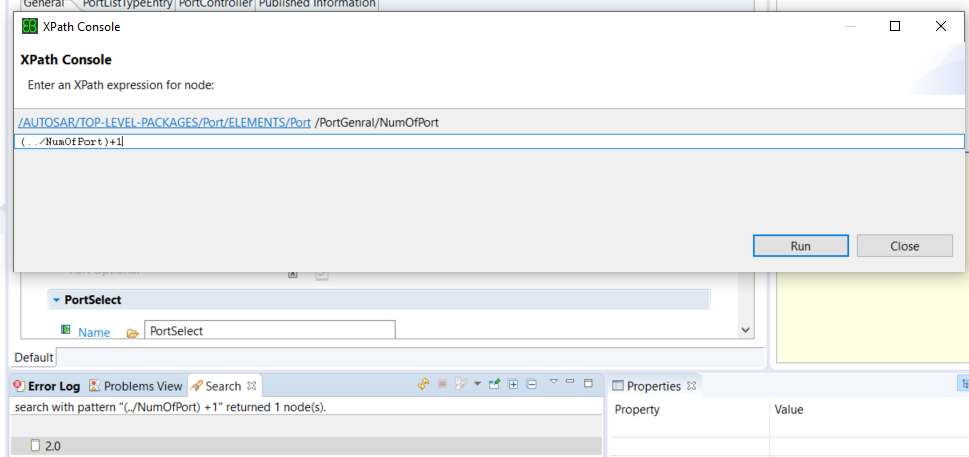








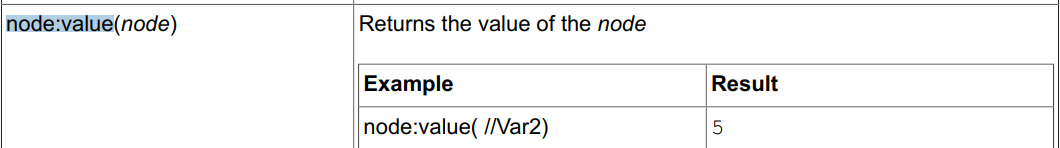
Ví dụ: Giá trị của một Node cộng với một giá trị bất kì



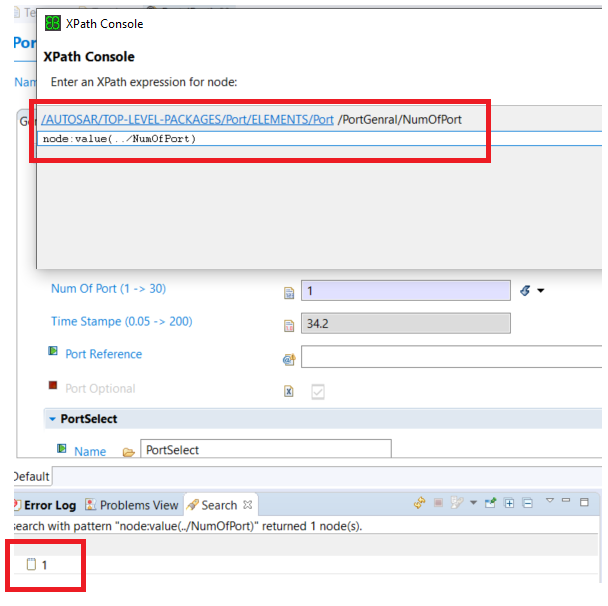
*d. XPath functions*

XPath chứa một tập hợp nhỏ các hàm dựng sẵn. Giúp cho việc lấy giá trị hay tìm kiếm một node một cách dễ dàng. Ở đây, mình chỉ giới thiệu một số hàm thường dùng nhất trong EB tresos. Để tìm hiểu thêm có thể truy cập tài liệu (2.4\_Studio\_documentation\_developers\_guide.pdf).

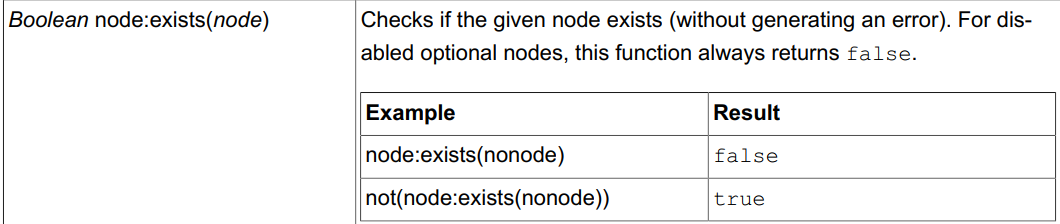
- **node:value(node) :** Trả về giá trị của một node



Ví dụ:

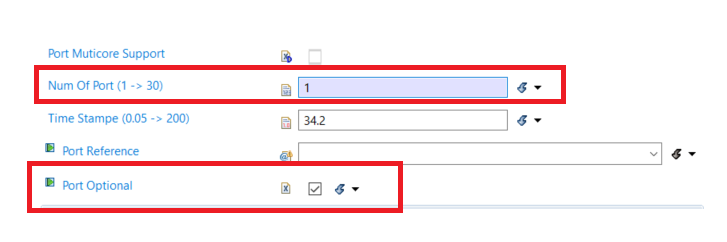


**-** *Boolean* **node:exists(**node**):** Kiểm tra có tồn tại một node không. Nếu tồn tại trả về true và ngược lại.

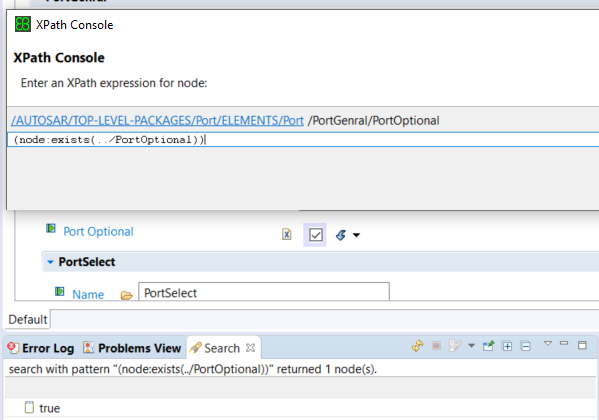


Ví dụ: Kiểm tra sự tồn tại của một node “PortOptional” thì node “NumOfPort” sẽ được “EDITABLE”.

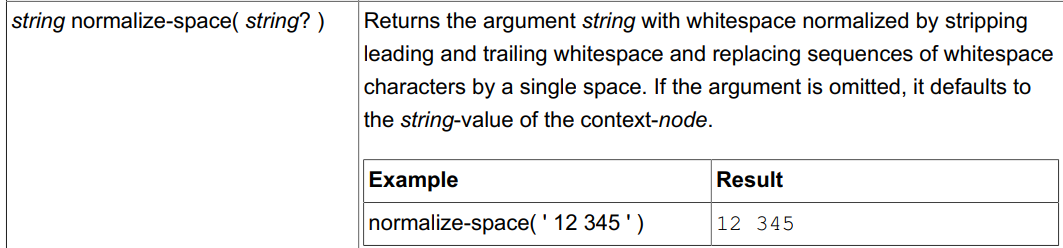




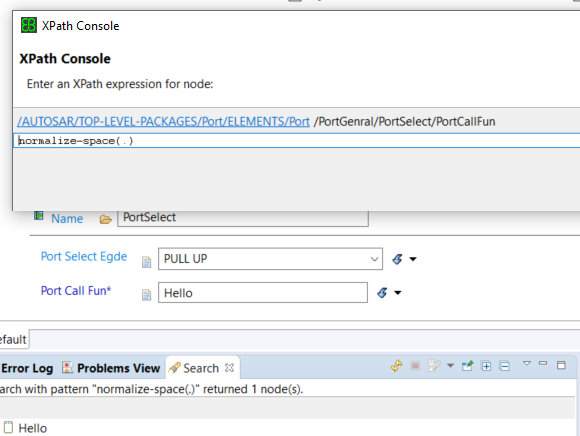
Chúng ta cũng có thể kiểm tra trên XPath Console.



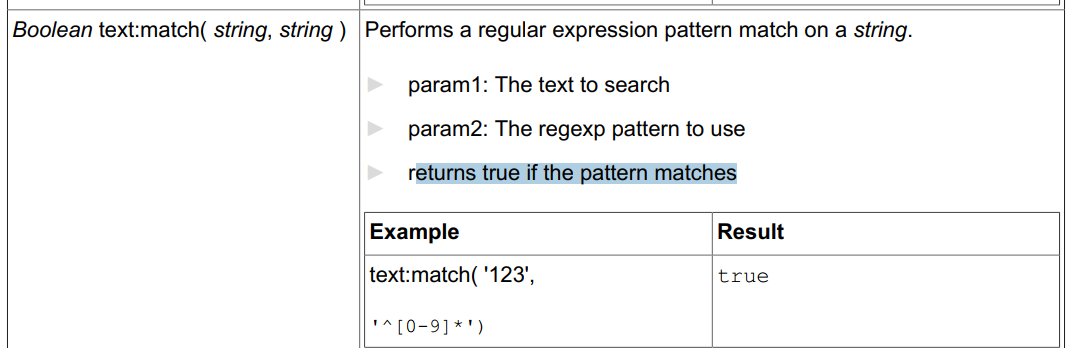
- *string* **normalize-space**( string? ): Trả về chuỗi đối số với khoảng trắng được chuẩn hóa bằng cách loại bỏ thay thế và theo dõi khoảng trắng và thay thế chuỗi khoảng trắng các ký tự bằng một khoảng trắng. Nếu đối số bị bỏ qua, nó sẽ mặc định là giá trị chuỗi của nút ngữ cảnh.



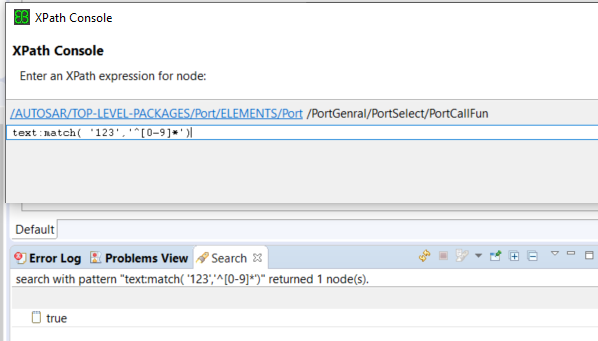
Ví dụ:



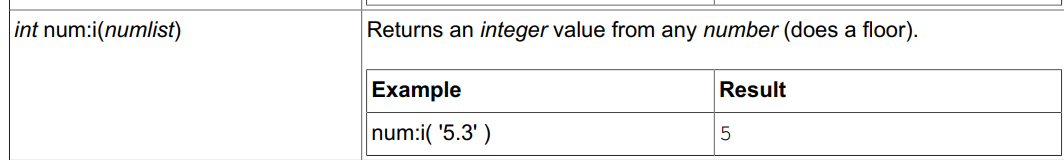
- *Boolean* **text:match**( string, string ): Trả về true nếu nằm trong khoảng phù hợp (regex pattern).



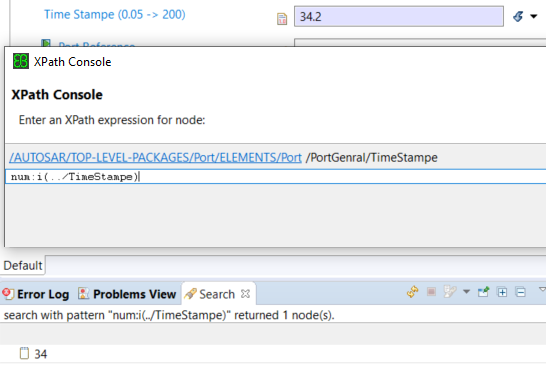
Ví dụ: Kiểm tra chuỗi ‘123’ có nằm trong phạm vi từ 0 đến 9 hay không. Trả về true nếu nằm trong mẫu (regex patternregex pattern) đó.



- *int* **num:i**(numlist): trả về một giá trị số nguyên từ bất kì số nào.

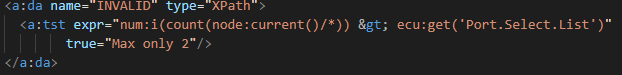


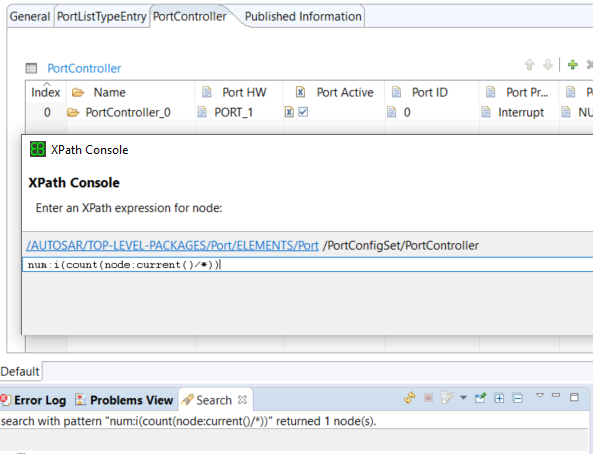
Ví dụ: Lấy giá trị số nguyên từ node với type=”FLOAT”.



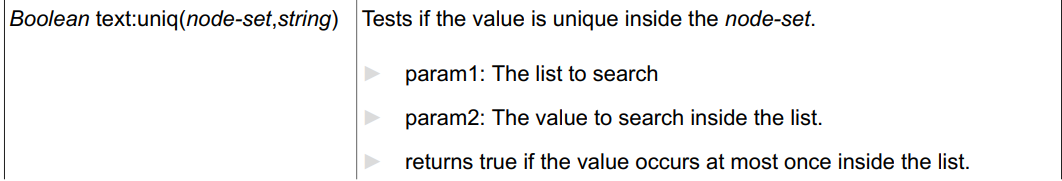
- *number* **count**(node-set): Trả về số lượng node trong tập hợp node đã cho. Hàm này thường đucợ sử dụng với **num:i** để lấy giá trị nguyên.

Ví dụ: Đếm có bao nhiêu node “PortController” trong list.

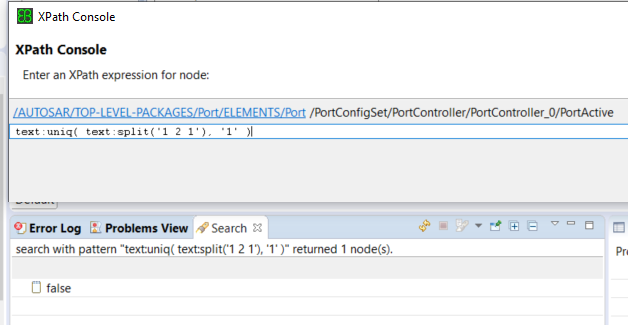




- *Boolean* **text:uniq**(node-set,string): Kiểm tra xem giá trị có phải là duy nhất bên trong tập hợp node hay không.

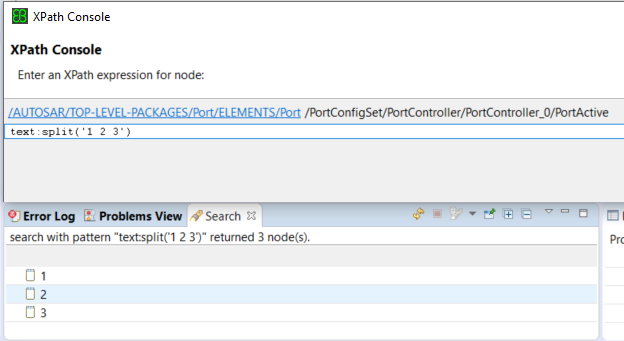


Ví dụ: Kiểm tra xem giá trị 1 có phải là duy nhất hay không. Nếu có hai giá trị 1 trong một tập hợp thì trả về false.



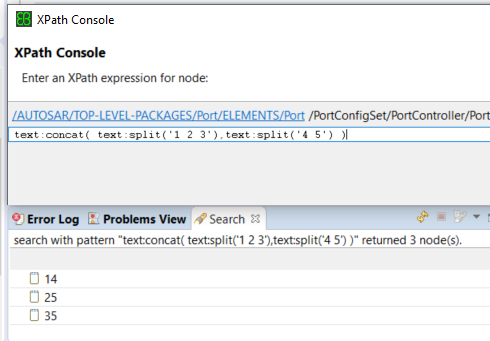
- *strlist* **text:split**(object): Tách biểu diễn chuỗi đối tượng đã cho thành một tập hợp các node.

Ví dụ: Tách chuỗi (‘1 2 3’).



*-strlist* **text:concat**(node-set, nodeset): Nối biểu diễn chuỗi của các đối tượng đã cho. Đầu tiên phần tử của tập hợp node được trả về là phép nối của các phần tử đầu tiên của cả hai tập hợp nút đã cho hai tập nút ngắn hơn cái kia, phần tử cuối cùng được sử dụng cho các tính toán mở. Nếu một trong tại ranh giới không gian.

Ví dụ: có hai node-set, A= text:split('1 2 3'), B= text:split(' 4 5')



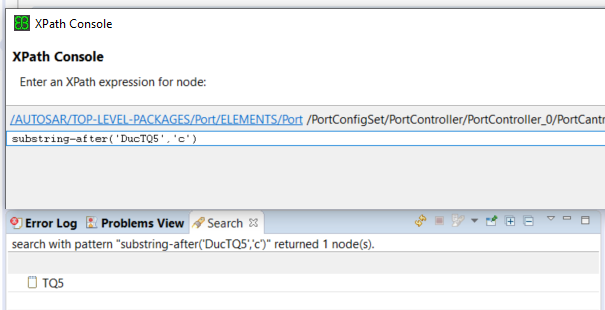
- *Boolean* **text:contains**(node-set,value): Trả về true nếu danh sách phần tử đã cho chứa giá trị đã cho.

Ví dụ: Kiểm tra trong list A=‘1 2 3’ có chứ giá trị 1 hay không.



- *string* **substring-after**(string, string): Trả về môt chuỗi mới với điều kiện lấy sau một kí tự bất kì trong chuỗi.

Ví dụ: Chuỗi A=’DucTQ5”, giờ chúng ta chỉ lấy giá trị sau kí tự ‘c’.



Ngoài ra còn có hàm **substring-before** thì ngược lại nhé.

- *string* **text:toupper**(string): Chuyển từ viết thường sang in hoa



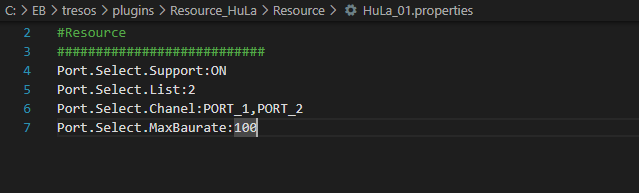
**2.2.5. ECU Resource Manager API**

*a. Registering ECU resource properties*

ECU resources được mô tả dưới dạng Java property file. Các file này lưu trữ các cặp key/value theo format sau:

<key>:<value>

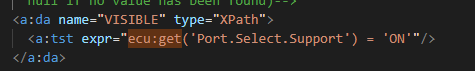
Key: Phải là duy nhất, không trùng lặp, trong file này sử dụng “#” để comment

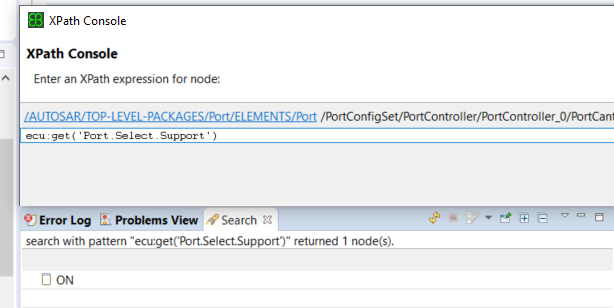


*b. ECU Resource XPath Functions*

- **ecu:get**(<key>): Trả về giá trị của một key tìm thấy trong ECU Resource hoặc nếu không tìm thấy sẽ trả về Null

Ví dụ: Một node lấy giá trị từ ECU Resource. Nếu giá trị lấy về khác ON thì node sẽ bị ẩn.

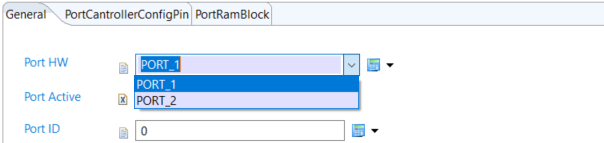


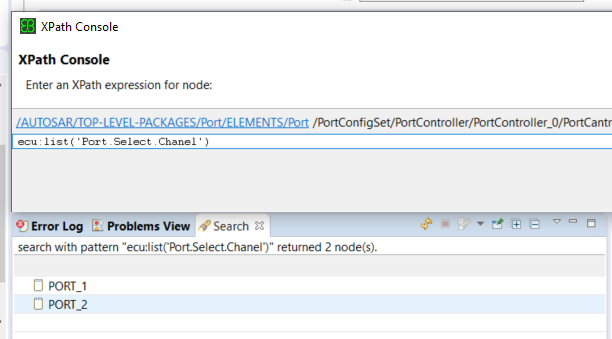


- **ecu:list**(<key>): Tách giá trị của key đã cho bằng dấu phẩy trong giá trị và trả về một tập hợp node.

Ví dụ: Giới hạn của một node nằm trong ECU Resource.







\* *Lưu ý: Khi chúng ta update một Resource thì nhớ tắt phần mềm EB tresos rôi bật lại nhé. Để cho phần mềm update lại Resource.*

**2.2.6. Template-based Code Generator API**

*a. Format*

Code generator lấy input stream, thực hiện mở rộng macro và ghi kết quả vào output stream. Tất cả các macro được Code generator sử dụng đều được tham số hóa bởi dữ liệu cấu hình. Các macro có thể truy cập vào dữ liệu với một biểu thức XPath cho phép chúng thực hiện các đánh giá toán học, liệt kê và thao tác chuỗi, giá trị kiểu Boolean.

Code template được đọc bởi code generator đơn giản là ASCII-file. Tất cả các code generate cần phải đặt trong tag:

[!CODE!]

...

[!ENDCODE!]

Nếu mà các đoạn code không nằm trong tag này thì khi gen sẽ không có.

Các biểu thức phải được đặt trong [!"<Expression>"!]

*b. Biến và Marco*

Biến trong code generators cho phép tạo ra các biến bên trong code template để sử dụng trong các biểu thức XPath. Các biến luôn được lưu trữ và diễn giải dưới dạng giá trị chuỗi.

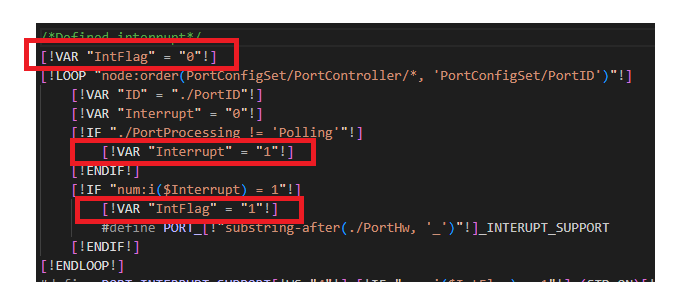
[!VAR "<Variable-Name>" = "<Expression>"!]

[!VAR "<Variable-Name>"!]

<Expression>

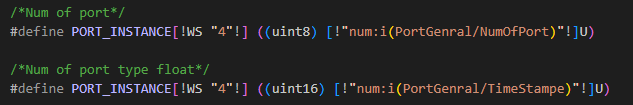
[!ENDVAR!]

Ví dụ: Sử dụng biến để lưu trữ giá trị từ một node trên EB tresos.

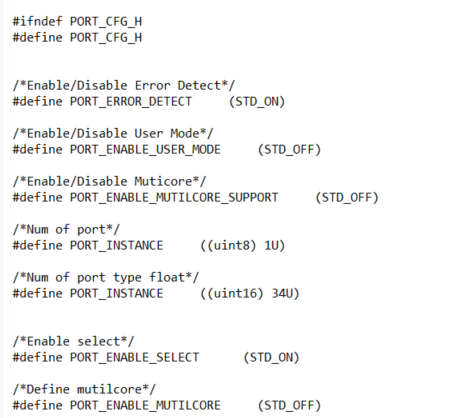


Marco: Tương tự như ngôn ngữ C, chúng ta cũng sẽ tạo ra một marco và giá trị của marco sẽ phụ thuộc vào một biểu thức XPath hay từ Resource,…

Ví du:



Sau khi chúng ta generator trên EB tresos.



*c.Câu lệnh điều kiện If trong Code template*

Các cấu trúc if là các biểu thức điều kiện được biết đến trong hầu hết các ngôn ngữ lập trình. Ở trong code template hoàn toàn giống với những ngôn ngữ khác.

Về Syntax:

!IF "<Expression>"!]

...

[!ELSEIF "<Expression>"!]

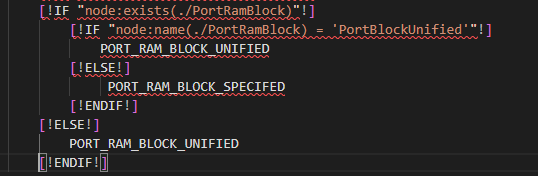
...

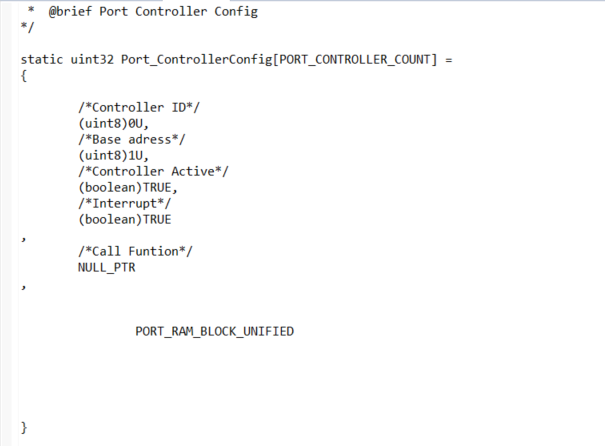
[!ELSE!]

...

[!ENDIF!]

Ví dụ: Nếu thoải mãn điều kiện thì sẽ generator " PORT\_RAM\_BLOCK\_UNIFIED” và ngược lại “PORT\_RAM\_BLOCK\_SPECIFED”





*d.Vòng lặp* LOOP

Ngôn ngữ XPath cho phép chọn một tập hợp các node, không chỉ một giá trị duy nhất.

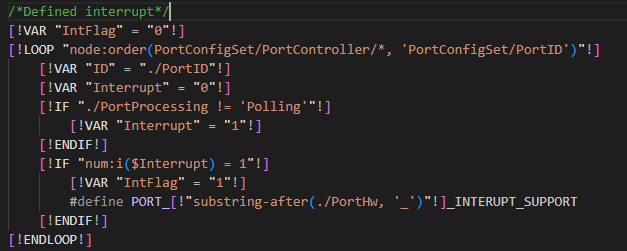
Code generator hỗ trợ lặp qua các tập hợp node đó. Trong khi lặp lại, con trỏ nút hiện tại được đặt một lần tới mỗi node của tập hợp node đã chọn. Cú pháp của một phép lặp là:

[!LOOP "<Expression>"!]

...

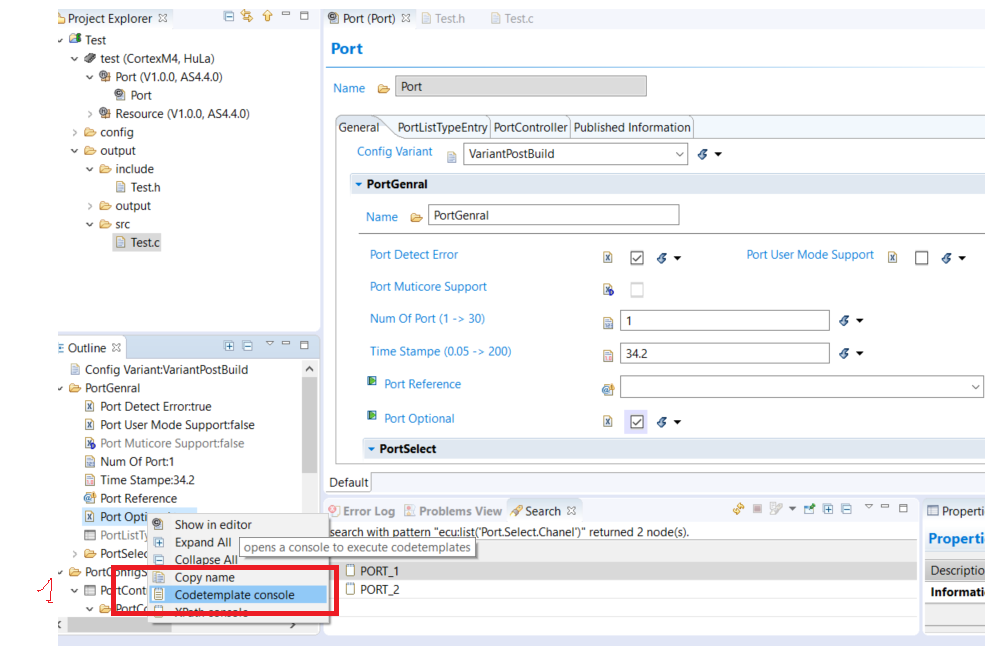
[!ENDLOOP!]

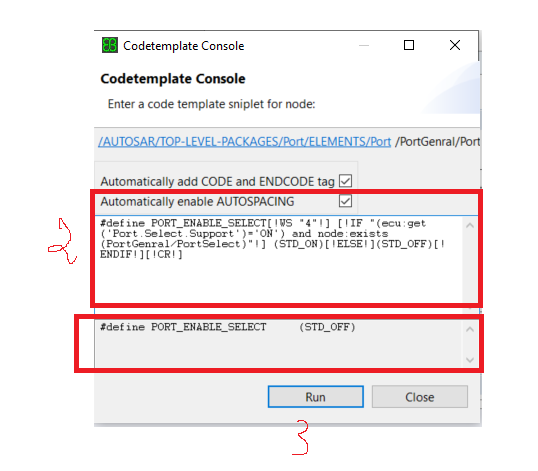
Ví dụ: Sử dụng vòng lặp để gen hết các trường hợp nhiều node ‘PortController’ trong list.



*e. Codetemplate console*

Tương tự như XPath console thì ở phần này chúng ta cũng có thể Debug để xem giá trị. Đầu tiên kích chuột phải chọn **Codetemlate console** như hình bên dưới. Sau đó nhập biểu thức mình muốn xem. Với việc sử dụng **Codetemlate console** chúng chúng ta đỡ tốn thời gian gen trên EB nhiều lần.





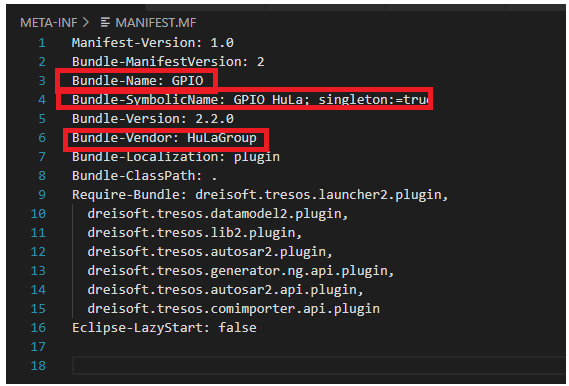
**Chương 3. How to create new module**

Để tạo một module mới chúng ta tiến hành theo các bước sau:

**3.1. Điều chỉnh MANIFEST.MF**

Thay đổi các mục sau trong META-INF/MANIFEST:

* Bundle-Name: Human-readable name for your plug-in.
* Bundle-SymbolicName: Unique name for your plug-in - a good practice is to use the directory-name of the plugin.
* Bundle-Vendor: the name of the vendor of the plugin



**3.2. Điều chỉnh plugin.xml**

Plugin.xml là một file được sử dụng để đăng kí resource với EB tresos Studio. Đăng kí được nhóm được gọi là extenions.

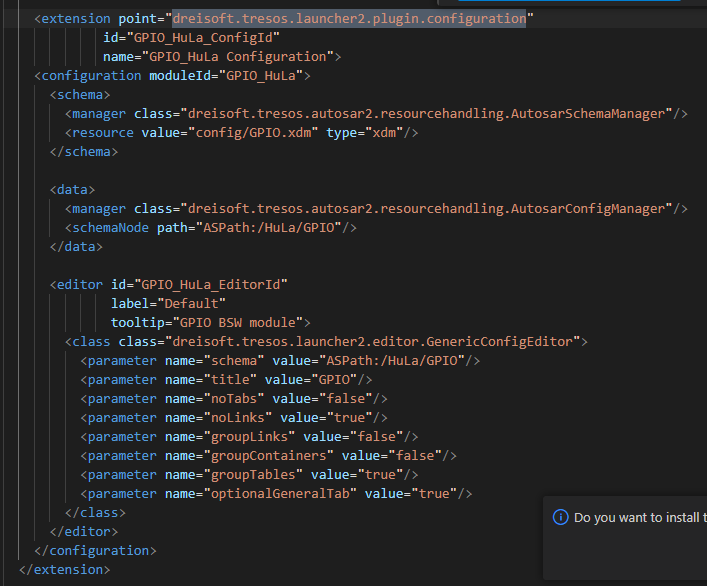
**3.2.1. dreisoft.tresos.launcher2.plugin.module**

Extension **dreisoft.tresos.launcher2.plugin.module** để đăng kí module mới.



**3.2.2. dreisoft.tresos.launcher2.plugin.configuration**

Extension **dreisoft.tresos.launcher2.plugin.configuration** dùng để đăng configuration specific resources cho module.



**3.2.3. dreisoft.tresos.launcher2.plugin.generator**

Extention **dreisoft.tresos.launcher2.plugin.generator** đăng kí code generator cho module.

