# Tableau récapitulatif de requêtes en XPath possibles dans Calames

# représentant les cas les plus courants

# pour faciliter les exports

Rappel : l'XPath est un langage de requête qui permet d'extraire des éléments d'un ou plusieurs fichiers xml.

Table des matières

[Tableau récapitulatif de requêtes en XPath possibles dans Calames 1](#_Toc156917717)

[représentant les cas les plus courants 1](#_Toc156917718)

[pour faciliter les exports 1](#_Toc156917719)

[XPath sur LEVEL de <archdesc> 2](#_Toc156917720)

[XPath sur <corpname> du <repository> 3](#_Toc156917721)

[XPath sur <unittitle> du <archdesc> 3](#_Toc156917722)

[XPath sur <origination> du <archdesc> 4](#_Toc156917723)

[XPath sur indexation en <controlaccess> du <archdesc> 4](#_Toc156917724)

[XPath sur un élément EAD hors du <did> du <archdesc> 6](#_Toc156917725)

[XPath sur critères d’arborescence au sein du fichier EAD 6](#_Toc156917726)

[XPath sur ID de <c> 9](#_Toc156917727)

[XPath sur LEVEL de <c> 10](#_Toc156917728)

[XPath sur contenu du <c> : <unitid> et <unittitle> 11](#_Toc156917729)

[XPath sur contenu du <c> : <unitdate> 13](#_Toc156917730)

[XPath sur <persname>, <corpname>, <famname>, <subject> et <geogname> du <c> 15](#_Toc156917731)

[XPath sur <genreform> du <c> 17](#_Toc156917732)

[XPath sur <physfacet> du <c> 18](#_Toc156917733)

[XPath sur <dao> du <c> 19](#_Toc156917734)

[XPath sur <scopecontent> du <c> 20](#_Toc156917735)

[XPath sur plusieurs éléments croisés du <c> 20](#_Toc156917736)

[Quelques principes de construction 21](#_Toc156917737)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objectif de l’export** | **XPath** | **Valide pour les exports** | | **Remarque** |
| XPath sur LEVEL de <archdesc> | | | | | | |
| Exporter tous les <unittitle> de <archdesc> lorsque le <archdesc> porte un LEVEL « fonds » | //archdesc[@level=**''fonds'']/did/unittitle** | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | N’a de sens que si l’export concerne plusieurs fichiers EAD simultanément (Le résultat de l’export donne autant de fichiers de sortie que de fichiers EAD sur lesquels l’export a été réalisé.)  En DublinCore simple ou qualifié au format xml et en MARC-xml, le <unittitle> est accompagné d’autres éléments présents dans le <archdesc>  A condition d’avoir un ID complété au niveau de l’archdesc. Si l’ID n’est pas rempli, l’export csv est décalé pour la première ligne. | | |
| XPath sur <corpname> du <repository> | | | | | | |
| Exporter tous les <did> de <archdesc> contenant « Bibliothèque de paléontologie » en PcData du <corpname> du <repository> | //archdesc/did**[repository/corpname[contains(text(),''Bibliothèque de paléontologie'')]]** | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | | N’a de sens que si l’export concerne plusieurs fichiers EAD simultanément  Un fichier résultant de l’export par fichier EAD sur lesquels l’export est réalisé  Pour le même XPath sur le NORMAL du <corpname> du <repository>, remplacer text() par @normal : //archdesc/did**[repository/corpname[contains(@normal,''Bibliothèque de paléontologie'')]]** |
| Exporter tous les <unittitle> de <archdesc> dont le <corpname> du <repository> contient « Bibliothèque de paléontologie » en PcData | //archdesc/did[repository/corpname[contains(text()**,''Bibliothèque de paléontologie''**)]]/unittitle | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | | N’a de sens que si l’export concerne plusieurs fichiers EAD simultanément  Un fichier résultant de l’export par fichier EAD sur lesquels l’export est réalisé  Pour le même XPath sur le NORMAL du <corpname> du <repository>, remplacer text() par @normal : //archdesc/did[repository/corpname[contains(@normal**,''Bibliothèque de paléontologie''**)]]/unittitle |
| XPath sur <unittitle> du <archdesc> | | | | | | |
| Exporter tous les <did> de <archdesc> contenant un <unittitle> contenant en PcData le mot « Laboratoire » | //archdesc/did**[unittitle[contains(text(),''Laboratoire'')]]** | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | N’a de sens que si l’export concerne plusieurs fichiers EAD simultanément  Un fichier résultant de l’export par fichier EAD sur lesquels l’export est réalisé | | |
| XPath sur <origination> du <archdesc> | | | | | |
| Exporter tous les <did> de <archdesc> contenant un <corpname> de ROLE « producteur » et au NORMAL « **Molécules de communication et adaptation des microorganismes (Paris ; 2011-....) »** | //archdesc/did**[\*[not(self::did)]//persname[@role=''producteur'']/@normal='' Dufour, Annie-Hélène (1947-2002)'']** | Tous les exports avec filtre requis. | N’a de sens que si l’export concerne plusieurs fichiers EAD simultanément  Un fichier résultant de l’export par fichier EAD sur lesquels l’export est réalisé | | |
| XPath sur indexation en <controlaccess> du <archdesc> | | | | | | |
| Exporter tous les <unittitle> de <archdesc> lorsqu’un <subject> portant le NORMAL « Poissons » est présent en <controlaccess> dans le <archdesc> | //archdesc[**controlaccess/**subject[@normal=**''**Poissons**'']]**/did/unittitle | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | Bien respecter la casse de la valeur d’attribut NORMAL  N’a de sens que si l’export concerne plusieurs fichiers EAD simultanément  Un fichier résultant de l’export par fichier EAD sur lesquels l’export est réalisé  Pour le même XPath sur l’AUTHFILENUMBER d’un <subject> lié à IdRef, remplacer @normal par @authfilenumber : //archdesc[**controlaccess/**subject[@authfilenumber=**''**027342700**'']]**/did/unittitle | | |
| Exporter tous les <unittitle> de <archdesc> lorsqu’un <genreform> portant le NORMAL « lois et règlements » est présent en <controlaccess> dans le <archdesc> | //archdesc[**controlaccess/genreform**[@normal=**''**lois et règlements**'']]**/did/unittitle | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | Bien respecter la casse de la valeur d’attribut NORMAL  N’a de sens que si l’export concerne plusieurs fichiers EAD simultanément  Un fichier résultant de l’export par fichier EAD sur lesquels l’export est réalisé  Pour le même XPath sur le TYPE du <genreform>, remplacer @normal par @type : //archdesc[**controlaccess/genreform**[@type=**''technique'']]**/did/unittitle | | |
| Exporter tous les <unittitle> de <archdesc> lorsqu’aucun <genreform> ne porte le NORMAL « texte imprimé » dans le <controlaccess> du <archdesc> | //archdesc[**controlaccess/genreform**[@normal!=**''texte imprimé'']]**/did/unittitle | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | Bien respecter la casse de la valeur d’attribut NORMAL  N’a de sens que si l’export concerne plusieurs fichiers EAD simultanément  Un fichier résultant de l’export par fichier EAD sur lesquels l’export est réalisé  Pour le même XPath sur le TYPE du <genreform>, remplacer @normal par @type : //archdesc[**controlaccess/genreform**[@type!=**''technique'']]**/did/unittitle | | |
| XPath sur un élément EAD hors du <did> du <archdesc> *L’XPath ci-dessous est valable pour les éléments directement sous l’élément <archdesc>, hors <did> et hors <dsc> : <acqinfo>,*  *<accessrestrict>, <userestrict>,…* | | | | | |
| Exporter tous les <unittitle> de <archdesc> lorsque le <acqinfo> du <archdesc> contient le mot « Don » dans un <p> | //archdesc[acqinfo/p[contains(text(),**''**Don**''**)]]/did/unittitle  Variante pour les <unittitle> de <archdesc> dont le <acqinfo> contient les mots « Don » ou « Legs » dans un <p> : //archdesc[acqinfo/p[contains(text(),**''**Don**''**)]]/did/unittitle | //archdesc[acqinfo/p[contains(text(),**''**Legs**''**)]]/did/unittitle | Tous les exports avec filtre requis. | N’a de sens que si l’export concerne plusieurs fichiers EAD simultanément  Un fichier résultant de l’export par fichier EAD sur lesquels l’export est réalisé | | |
| XPath sur critères d’arborescence au sein du fichier EAD | | | | | | |
| Exporter les <c> de premier niveau et leurs enfants | //dsc/c | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | | Sur un ou plusieurs fichiers EAD à la fois  Lors d’un export DublinCore-CSV-Filtre, les enfants des <c> de premier niveau ne sortent pas en résultat. |
| Exporter les <c> de 2e niveau et leurs enfants. | //dsc/descendant-or-self::c[../../../dsc] | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | | Sur un ou plusieurs fichiers EAD à la fois. Si plusieurs fichiers à la fois, bien s’assurer que chaque fichier dispose bien de deux niveaux de <c>  Ajouter ../ pour chaque niveau de <c> supplémentaire au sein du fichier EAD :  **//dsc/descendant-or-**self::c[../../../../dsc] sélectionne par exemple les <c> de 3e niveau ;  //dsc/descendant-or-self::c[../../../../../dsc] sélectionne par exemple les <c> de 5e niveau, etc… |
| Exporter le 4e <c> de premier niveau dans l’ordre de la fratrie des <c> de premier niveau au sein du ou des fichiers EAD concerné(s) et ses enfants. | **//dsc/c[position()=4]** | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | | Sur un ou plusieurs fichiers EAD à la fois. Si plusieurs fichiers à la fois, bien s’assurer que chaque fichier dispose bien d’au moins quatre <c> de 1er niveau  Lors d’un export DublinCore-CSV-Filtre, les enfants du <c> sélectionné ne sortent pas en résultat. Pour obtenir également ses enfants en résultat de l’export DublinCore-CSV-Filtre, saisir : **//dsc/c[position()=4]**/descendant-or-self::c  Ajouter /c pour chaque niveau de <c> supplémentaire enfant du premier <c> de chaque niveau au sein du fichier EAD :  **//dsc/c/c[position()=3] pour le <c> de 2e niveau en 3e position dans sa fratrie et enfant du 1er <c> de 1er niveau //dsc/c/c/c[position()=2] pour le <c> de 3e niveau en 2e position dans sa fratrie et enfant du 1er <c> de 1er niveau et du 1er <c> de 2e niveau** |
| Exporter le 4e <c> de 3e niveau enfant du 1er <c> de 1er niveau et du 7e <c> de 2e niveau et ses enfants | **//dsc/c/c[position()=7]/c[position()=4]** | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | | Sur un ou plusieurs fichiers EAD à la fois. Si plusieurs fichiers à la fois, bien s’assurer que chaque fichier dispose bien du bon nombre de <c> à chaque niveau  Lors d’un export DublinCore-CSV-Filtre, les enfants des <c> sélectionnés ne sortent pas en résultat |
| Exporter tous les <c> suivant au sein de la même fratrie le <c> dont l’ID est saisi et leurs enfants | //c[@id=''FRPALME0000000000018-1-1-25-1'']/following-sibling::\* | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | | Sur un seul fichier EAD à la fois.  Le <c> dont l’ID est saisi ne sort pas en résultat de cet export.  Lors d’un export DublinCore-CSV-Filtre, les enfants des <c> sélectionnés ne sortent pas en résultat |
| Exporter tous les <c> précédant au sein de la même fratrie le <c> dont l’ID est saisi, ainsi que leur parent et leurs enfants | //c[@id=''Calames-201491159303861'']/preceding-sibling::\* | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | | Sur un seul fichier EAD à la fois.  Le <c> dont l’ID est saisi ne sort pas en résultat de cet export.  Lors d’un export DublinCore-CSV-Filtre, les parents et les enfants des <c> sélectionnés ne sortent pas en résultat |
| Exporter les 10 <c> suivant immédiatement au sein de la même fratrie le <c> dont l’ID est saisi et leurs enfants | //c[@id=''FRPALME0000000000018-1-1-42-1'']/following-sibling::c[position()&lt;10] | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | | Sur un seul fichier EAD à la fois.  *&lt; est ici employé pour <*  Voir les caractères de remplacement pour éviter les caractères parasites ici [lien]  Lors d’un export DublinCore-CSV-Filtre, les enfants des 10 <c> sélectionnés ne sortent pas en résultat. |
| Exporter tous les <c> n’ayant aucun enfant | **//dsc/descendant-or-self::c[not(child::c)]** | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | | Sur un ou plusieurs fichiers EAD à la fois.  La requête exporte aussi les parents. |
| XPath sur ID de <c> | | | | | | |
| Exporter un <c> et ses enfants à partir de son ID | //c[@id=''Calames-2015521155834512''] | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | | Sur un ou plusieurs fichiers EAD à la fois.  Lors d’un export DublinCore-CSV-Filtre, les enfants du <c> dont l’ID est saisi ne sortent pas en résultat. Pour obtenir le niveau de <c> dont l’ID est saisi et tous ses sous-composants et leurs enfants lors d’un export DublinCore-CSV-Filtre, utiliser : //c[@id=''Calames-2015521155834512'']/descendant-or-self::\* |
| Exporter des <c> et leurs enfants à partir de leurs ID | **//c[@id=''id du 1er <c> à exporter''] | //c[@id=''id du 2e <c> à exporter''] | //c[@id=''id du 3e <c> à exporter'']** | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, MARC-xml, MARCISO2709, MARCISO2709UTF8 | | Sur un ou plusieurs fichiers EAD à la fois.  Cet XPath ne fonctionne pas pour l’export DublinCore-CSV-Filtre |
| **XPath sur LEVEL de <c>** | | | | | | |
| Exporter t**ous les <c> portant un level « recordgrp »** et leurs enfants | **//c[@level=''recordgrp'']** | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | Sur un ou plusieurs fichiers EAD à la fois.  Lors d’un export DublinCore-CSV-Filtre, les enfants ne sortent pas en résultat.  Pour sélectionner lors d’un même export les <c> portant des LEVEL différents, par exemple pour obtenir à la fois les <c> de LEVEL « recordgrp » et les <c> de LEVEL « item », saisir : **//c[@level=''recordgrp''] | //c[@level=''item'']** | | |
| Exporter t**ous les <c> ne portant pas un LEVEL « item »** et leurs enfants ne portant pas un LEVEL « item » | **//c[@level!=''item'']** | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | Sur un ou plusieurs fichiers EAD à la fois. | | |
| XPath sur contenu du <c> : <unitid> et <unittitle> | | | | | | |
| Exporter tous les <c> contenant un <unitid> de TYPE cote sans leurs enfants | **//c[did/unitid[@type=''cote'']]** | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | Sur un ou plusieurs fichiers EAD à la fois.  Pour obtenir également les enfants des <c> contenant un <unitid> de TYPE « cote » (ne fonctionne pas avec l’export DublinCore-CSV-Filtre) : //c**[\*[not(self::c)]//unitid[@type=''cote'']]/**descendant-or-self::c | | |
| Exporter les <c> contenant un <unitid> dont le PcData contient les deux caractères « Ms » sans leurs enfants | **//c[did/unitid[contains(text(),''Ms'')]]** | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | Sur un ou plusieurs fichiers EAD à la fois.  Pour obtenir également les enfants des <c> contenant un <unitid> de TYPE « cote » contenant « Ms » en PcData, saisir : //c**[\*[not(self::c)]//unitid[contains(text(),''Ms'')]]/**descendant-or-self::c | | |
| Exporter les <c> contenant un <unitid> dont le PcData commence par les deux caractères « Ms » sans leurs enfants | **//c[did/unitid[substring(text(),1,2)=''Ms'']]** | Tous les exports avec filtre requis.  Répétition des résultats avec l’export DublinCore-CSV-Filtre | Sur un ou plusieurs fichiers EAD à la fois.  Pour obtenir également les enfants des <c> contenant dans leur <unitid> dont le PcData commence par les deux caractères « Ms » : **//c[did/unitid[substring(text(),1,2)=''Ms'']]/**descendant-or-self::c | | |
| Exporter les <c> contenant un <unitid> de TYPE « cote » dont le PcData ne contient pas les caractères « Ms » sans leurs enfants | **//c[did/unitid[type=''cote''][not(contains(text(),''Ms''))]]** | Tous les exports avec filtre requis.  Répétition des résultats avec l’export DublinCore-CSV-Filtre | Sur un ou plusieurs fichiers EAD à la fois.  Pour obtenir également les enfants des <c> contenant un <unitid> de TYPE « cote » ne contenant pas « Ms » en PcData, saisir : **//c[did/unitid[type=''cote''][not(contains(text(),''Ms''))]]/**descendant-or-self::c  Pour obtenir la liste des <unitid> de TYPE « cote » ne contenant pas « Ms » en PcData (pour les exports Natif-C et EvalXpath uniquement) : **//unitid[type=''cote''][not(contains(text(),''Ms''))]** | | |
| Exporter les <c> contenant un <unittitle> qui contient un <title> portant un attribut ROLE titre sans leurs enfants | //c**[\*[not(self::c)]//unittitle/title[@role=''titre'']]** | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, MARC-xml, MARCISO2709, MARCISO2709UTF8 | Sur un ou plusieurs fichiers EAD à la fois.  Cet XPath ne fonctionne pas pour l’export DublinCore-CSV-Filtre  Pour obtenir également les enfants des <c> contenant dans leur <unittitle> un <title> de ROLE « titre » : //c**[\*[not(self::c)]//unittitle/title[@role=''titre'']]/**descendant-or-self::c | | |
| Exporter tous les <c> contenant un <unittitle> dont le PcData contient les caractères « Comptes rendus » sans leurs enfants | **//c[did/unittitle[contains(text(),''Comptes rendus'')]]** | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, DublinCore-CSV-Filtre, MARC-xml, MARCISO2709, MARCISO2709UTF8 | Sur un ou plusieurs fichiers EAD à la fois.  Pour obtenir également les enfants des <c> contenant « Comptes rendus » dans leur <unittitle> en PcData (ne fonctionne pas avec l’export DublinCore-CSV-Filtre) : //c**[\*[not(self::c)]//unittitle[contains(text(),''Comptes rendus'')]]/**descendant-or-self::c | | |
| XPath sur contenu du <c> : <unitdate> | | | | | |
| Exporter les <c> comprenant des <unitdate> dont le NORMAL commence par les caractères « 191 » sans leurs enfants | **//c[\*[not(self::c)]//unitdate[substring(@normal,1,3)=''191'']]** | Tous les exports avec filtre requis.  Répétition des résultats avec l’export DublinCore-CSV-Filtre | Substring permet de recherche une chaîne de caractères dans le PCData ou un attribut d’un élément. Préciser la position des caractères dans le contenu du PCData ou de l’attribut interrogés entre parenthèses après substring. Ici, les caractères 1 à 3 du NORMAL **(@normal,1,3).**  Pour obtenir également les enfants des <c> contenant un <unitdate> dont le NORMAL contient pour trois premiers caractères « 191 » : **//c[\*[not(self::c)]//unitdate[substring(@normal,1,3)=''191'']]/descendant-or-self::c** | | |
| Exporter les <c> comprenant des <unitdate> où qu’il se trouve dans le <c> dont le NORMAL est 1914 sans leurs enfants | **//c[\*[not(self::c)]//unitdate[@normal=''1914'')]]** | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, MARC-xml, MARCISO2709, MARCISO2709UTF8 | Sur un ou plusieurs fichiers EAD à la fois.  Cet XPath ne fonctionne pas pour l’export DublinCore-CSV-Filtre | | |
| Exporter les <c> comprenant un <unitdate> dont les quatre premiers caractères du NORMAL sont supérieurs à « 1990 » | **//c[\*[not(self::c)]//unitdate[substring(@normal,1,4)**&gt;**''1990'']]** | Tous les exports avec filtre requis.  Répétition des résultats avec l’export DublinCore-CSV-Filtre | Sur un ou plusieurs fichiers EAD à la fois.  Pour exprimer supérieur ou inférieur à, voir la [liste des caractères spéciaux](https://documentation.abes.fr/aidecalames/manuelcorrespondant/index.html#ExportModele_2:yZqkH26Dd3dlAeifhovSOb) valable également pour les XPath | | |
| Exporter les <c> héritant d’un <unitdate> portant 1914 en NORMAL sans leurs enfants | //c[ancestor::c/\*[not(self::c)]//**unitdate[@normal=''1938/1939'']]**  **Variante pour exporter** les <c> héritant d’un <unitdate> dont le NORMAL commence par les caractères « 193 » : //c[ancestor::c/\*[not(self::c)]//**unitdate[substring(@normal,1,3)=''193'']]** | Tous les exports avec filtre requis.  Répétition des résultats avec l’export DublinCore-CSV-Filtre | Sur un ou plusieurs fichiers EAD à la fois. | | |
| XPath sur <persname>, <corpname>, <famname>, <subject> et <geogname> du <c> *Les XPath ci-dessous sont valables pour les <persname>, mais peuvent être adaptés en remplaçant <persname> par <corpname> ou <famname>. Leur construction est aussi valable pour les <subject>, <geogname>. Lorsque des ajustements sont nécessaires, ils sont précisés.* | | | | | |
| Exporter tous les <c> contenant un <persname> portant un ROLE « auteur » et leurs enfants | //persname[@role=**''**070**''**]/ancestor::c[1] | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, MARC-xml, MARCISO2709, MARCISO2709UTF8 | Sur un ou plusieurs fichiers EAD à la fois.  Cet XPath ne fonctionne pas pour l’export DublinCore-CSV-Filtre  Les <c> peuvent sortir plusieurs fois s’ils sont enfants d’un <c> contenant un <persname> ROLE « auteur » et contiennent eux-mêmes un <persname> ROLE « auteur ».  Pour la liste des codes fonctions UNIMARC utilisés dans Calames pour les ROLE des <persname>, se reporter à la [documentation Calames](https://documentation.abes.fr/aidecalames/manuelcatalogageead/index.html#CFPRole) | | |
| Exporter tous les <c> contenant un <persname> portant un ROLE « auteur » et dont le NORMAL est « **Curie, Ève (1904-2007) » sans leurs enfants** | **//c[\*[not(self::c)]//persname[@role=''070''][@normal=''Curie, Ève (1904-2007)'')]]** | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, MARC-xml, MARCISO2709, MARCISO2709UTF8 | Sur un ou plusieurs fichiers EAD à la fois.  Cet XPath ne fonctionne pas pour l’export DublinCore-CSV-Filtre  Pour obtenir également les enfants des <c> contenant un <persname> de ROLE « auteur » et dont le NORMAL est « **Curie, Ève (1904-2007) » :**  **//c[\*[not(self::c)]//persname[@role=''070''][@normal=''Curie, Ève (1904-2007)'']]/**descendant-or-self::c | | |
| Exporter tous les <c> contenant un <persname> portant un ROLE « auteur » et dont le nom de famille en NORMAL est « Curie » **sans leurs enfants** | **//c[\*[not(self::c)]//persname[@role=''070''][substring-before(@normal,'','')=''Curie'']]**  Variante pour obtenir les <c> contenant un <subject> sur un modèle proche : **//c[\*[not(self::c)]//subject[contains(@normal,''Cuivre'')]]** | Tous les exports avec filtre requis.  Répétition des résultats avec l’export DublinCore-CSV-Filtre | Sur un ou plusieurs fichiers EAD à la fois.  Pour obtenir également les enfants des <c> contenant un <persname> de ROLE « auteur » et dont le nom de famille en NORMAL est « **Curie »** :  **//c[\*[not(self::c)]//persname[@role=''070''][substring-before(@normal,'','')=''Curie'']]/**descendant-or-self::c  Pour obtenir la liste des NORMAL de <subject> (pour les exports Natif-C et EvalXpath uniquement) ://subject/@normal | | |
| Exporter les <c> contenant un <subject> où qu’il se trouve dans le <c> et dont le PcData contient « Squaliformes » sans leurs enfants | //c**[\*[not(self::c)]//subject[contains(text(),''Squaliformes'')]]** | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, MARC-xml, MARCISO2709, MARCISO2709UTF8 | Sur un ou plusieurs fichiers EAD à la fois.  Cet XPath ne fonctionne pas pour l’export DublinCore-CSV-Filtre  Pour obtenir également les enfants des <c> contenant un <subject> qui contient « Squaliformes » en PcData : //c**[\*[not(self::c)]//subject[contains(text(),''Squaliformes'')]]/**descendant-or-self::c | | |
| Exporter tous les ID de <c> contenant un <persname> non lié à IdRef (également valable pour <corpname>, famname>, <subject> et <geogname> en remplaçant <persname> par l’un de ces éléments ; pour les <corpname>, il est normal que le <corpname> du <repository> ne soit pas lié à IdRef) | **//c[\*[not(self::c)]//persname[not(@source)]]/@id**  **Variante pour exporter** les ID de <c> contenant un <persname>, <corpname> ou famname> lié à IdRef (également valable pour <subject> et <geogname>) : **//c[\*[not(self::c)]//persname[not(@source)]]/id | //c[\*[not(self::c)]//corpname[not(@source)]]/id | //c[\*[not(self::c)]//famname[not(@source)]]/id** | Natif-C, EvalXpath | Sur un ou plusieurs fichiers EAD à la fois.  Pour exporter directement les <persname>, <corpname> et famname> (également valable pour <geogname> ou <subject>) non liés à IdRef (pour les exports Natif-C et EvalXpath uniquement) : **//persname[not(@source)] | //corpname[not(@source)] | //famname[not(@source)]** | | |
| Exporter tous les <c> contenant un <persname> portant un ppn donné en AUTHFILENUMBER sans leurs enfants (également valable pour <corpname>, famname>, <subject> et <geogname> en remplaçant <persname> par l’un de ces éléments, il est normal que le <corpname> du <repository> ne soit pas lié à IdRef) | **//c[\*[not(self::c)]//persname[@authfilenumber=''032577508'']]** | Tous les exports avec filtre requis.  Répétition des résultats avec l’export DublinCore-CSV-Filtre | Sur un ou plusieurs fichiers EAD à la fois.  Pour obtenir également les enfants des <c> contenant un <persname> portant en AUTHFILENUMBER le ppn **032577508 :**  **//c[\*[not(self::c)]//persname[@authfilenumber=''032577508'']]/**descendant-or-self::c  Pour obtenir la liste des AUTHFILENUMBER de <persname> (pour les exports Natif-C et EvalXpath uniquement) ://persname/@authfilenumber | | |
| XPath sur <genreform> du <c> | | | | | |
| Exporter les <c> héritant d’une indexation <genreform> « type de document » | //c[ancestor::c/\*[not(self::c)]//genreform[@type="type de document"]] | **Non testé** |  | | |
| Exporter les <c> héritant d’une indexation <genreform> portant « texte imprimé » en NORMAL | //c[ancestor::c/\*[not(self::c)]//genreform[@normal=**''**texte imprimé**'']]** | Tous les exports avec filtre requis.  Répétition des résultats avec l’export DublinCore-CSV-Filtre | Sur un ou plusieurs fichiers EAD à la fois. | | |
| XPath sur <physfacet> du <c> | | | | | |
| Exporter tous les <c> contenant un <physfacet> de TYPE « technique » sans leurs enfants | //c**[did/physdesc/**physfacet[@type=**''**technique**''**]] | Tous les exports avec filtre requis.  Répétition des résultats avec l’export DublinCore-CSV-Filtre | Sur un ou plusieurs fichiers EAD à la fois.  Pour obtenir également les enfants des <c> contenant un <physfacet> de TYPE « technique » : //c**[\*[not(self::c)]//**physfacet[@type=**''**technique**''**]]**/**descendant-or-self::c  Pour obtenir la liste des PcData de <physfacet> de TYPE « technique » (pour les exports Natif-C et EvalXpath uniquement) ://physfacet[@type=**''**technique**''**]/text() | | |
| Exporter tous les <c> contenant un <physfacet> de type technique dont le PcData contient les caractères « Dessins » | //c**[\*[not(self::c)]//**physfacet[@type=**''**technique**''**]/text()[contains(.,'Dessins')]] | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, MARC-xml, MARCISO2709, MARCISO2709UTF8 | Sur un ou plusieurs fichiers EAD à la fois.  Cet XPath ne fonctionne pas pour l’export DublinCore-CSV-Filtre | | |
| Exporter tous les <c> contenant un <physfacet> qui n’est pas de type « technique » sans leurs enfants | //c**[\*[not(self::c)]//**physfacet[@type!= **''**technique**''**]] | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, MARC-xml, MARCISO2709, MARCISO2709UTF8 | Sur un ou plusieurs fichiers EAD à la fois.  Cet XPath ne fonctionne pas pour l’export DublinCore-CSV-Filtre  Pour obtenir également les enfants des <c> contenant un <physfacet> qui n’est pas de type « technique » : //c**[\*[not(self::c)]//**physfacet[@type!= **''**technique**''**]]**/**descendant-or-self::c | | |
| Exporter tous les <c> dont le <physfacet> de type « conditionnement » contient « Entoilage », mais ne contient pas « Encadrement » en PcData sans leurs enfants | //c**[\*[not(self::c)]//**physfacet[@type=**''**conditionnement**''**]/text()[contains(.,**''**Entoilage**''**)][not(contains(.,**''**Encadrement**''**))]] | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, MARC-xml, MARCISO2709, MARCISO2709UTF8 | Sur un ou plusieurs fichiers EAD à la fois.  Cet XPath ne fonctionne pas pour l’export DublinCore-CSV-Filtre  Pour obtenir également les enfants des <c> contenant un <physfacet> de type « conditionnement » qui contient « Entoilage », mais ne contient pas « Encadrement » en PcData : //c**[\*[not(self::c)]//**physfacet[@type=**''**conditionnement**''**]/text()[contains(.,**''**Entoilage**''**)][not(contains(.,**''**Encadrement**''**))]]**/**descendant-or-self::c | | |
| XPath sur <dao> du <c> | | | | | |
| Exporter les <c> contenant un <dao> ou un <daogrp> et leurs enfants | **//dsc/descendant-or-self::c[dao | daogrp]** | Natif-C, Natif-Filtre,  DublinCore simple, DublinCore qualifié, MARC-xml, MARCISO2709, MARCISO2709UTF8 | | Sur un ou plusieurs fichiers EAD à la fois. |
| Exporter les <unitid> de tous les <c> contenant un HREF ou un ROLE de <dao> vide | **//c[dao[not(@href)]]/did/unitid | //c[dao[not(@role)]]/did/unitid** | Natif-C, EvalXpath | | Sur un ou plusieurs fichiers EAD à la fois. |
| Exporter la liste des TITLE de <dao> | **//dao/@title** | Natif-C, EvalXpath | | Sur un ou plusieurs fichiers EAD à la fois. |
| XPath sur <scopecontent> du <c> | | | | |
| Exporter les <c> contenant un <scopecontent> faisant état d’une mention manuscrite dans le document sans leurs enfants | //c**[scopecontent/p[contains(text(),''Mention manuscrite'')]]** | Tous les exports avec filtre requis.  Répétition des résultats avec l’export DublinCore-CSV-Filtre | | Sur un ou plusieurs fichiers EAD à la fois.  Pour obtenir également les enfants des <c> contenant « Mention manuscrite » dans leur <scopecontent> en PcData : //c**[scopecontent/p[contains(text(),''Mention manuscrite'')]]/**descendant-or-self::c |
| XPath sur plusieurs éléments croisés du <c> *Les XPath suivants ont pour but de ne donner en résultat qu’un élément du <c>, ils ne peuvent donc être effectués qu’avec l’export Natif-C* | | | | |
| Exporter les <unittitle> des <c> qui contiennent un <persname> de ROLE « auteur » | //c**[\*[not(self::c)]//**persname[@role=**''070'']]/did/unittitle** | Natif-C | | Sur un ou plusieurs fichiers EAD à la fois. |
| Exporter les <unittitle> des <c> qui contiennent un <geogname> contenant « Meuse » en PcData et un <unitdate> contenant « 1915 » en NORMAL | //c**[\*[not(self::c)]//geog**name[contains(text(),**''Meuse'')]] and //unitdate[contains(@normal,''1915'')]]/did/unittitle** | Natif-C | | Sur un ou plusieurs fichiers EAD à la fois.  Variante valable pour tous les exports pour exporter le contenu entier du <c> avec les mêmes critères de sélection : //c**[\*[not(self::c)]//geog**name[contains(text(),**''Meuse'')]] and //unitdate[contains(@normal,''1915'')]]** |
| Exporter les <unittitle> des <c> qui contiennent un <unitdate> dont le NORMAL est « 1924 » et un <unitid> de TYPE « ancienne\_cote » | //c**[\*[not(self::c)]//unitdate[@normal=''1924'']]/did[unitid[@type=''ancienne\_cote'']]/unittitle** | Natif-C | | Sur un ou plusieurs fichiers EAD à la fois.  Variante valable pour tous les exports pour exporter le contenu entier du <c> avec les mêmes critères de sélection : //c[did/**unitid[@type=''ancienne\_cote'']][\*[not(self::c)]//unitdate[@normal=''1924'']]** |
| Exporter les <unitid> des <c> qui contiennent des <physfacet> de TYPE « reliure » | //c/did[physdesc/physfacet[@type=**''reliure'']]/unitid** | Natif-C**,** EvalXpath | | Sur un ou plusieurs fichiers EAD à la fois. |

## Quelques principes de construction

- //: permet de trouver un élément EAD quelque soit la position dans l’arborescence XML

- balise/balise/balise : chemin complet de l’élément EAD dont on veut l’export

- si on demande des //c on aura des composants complets

- Entre crochets […] : la condition à remplir pour être retenu. Attention le contenu des crochets doit être adapté au niveau de la hiérarchie (élément EAD) dans lequel on s’est précédemment positionné

La condition peut porter

* Sur une valeur d’indicateur ; exemple physfacet[@type=''reliure''] : ne prend en compte que ces <physfacet> particulier
* Sur un contenu textuel (PCData) de l’élément EAD ; exemple corpname[contains(text(),''Bibliothèque de paléontologie'')]]
* Une chaine de caractère soit précisément située [substring(text(), position du 1re caractère à prendre en compte, nombre de caractères à prendre en compte ; exemple **unitid[substring(text(),1,2)=''Ms'']]**

**Soit située par rapport à un caractère précisé mais non situé : dans cet exemple “Curie” avant la virgule //c[\*[not(self::c)]//persname[@role=''070''][substring-before(@normal,'','')=''Curie'']]/**descendant-or-self::c

* Sur la position dans une fratrie : exemple c[position()=4]

La condition peut exprimer la présence ou au contraire l’absence **(**signe **!=**) :

* Valeur d’attribut : signe **=**  pour la présence ; signe **!=**  pour l’absence
* Balise : signe pour l’absence : not(nom balise ou contenu textuel) **//c[did/unitid[type=''cote''][not(contains(text(),''Ms''))]]**

On peut demander le noeud XML visé seul (par défaut) ou

* les noeuds suivants de même niveau **: /following-sibling::\***
* les nœuds précédents de même niveau **:** /preceding-sibling::\*
* le nœud visé et ses enfants : **/**descendant-or-self::\*
* un noeud ancêtre identifié par son nom de balise /ancestor::exemple//persname[*@role*=**''**070**''**]/ancestor::**c**[1]

On peut vouloir exclure les composants enfants

Le pipe | est utilisé pour l’opérateur booléen « OU » (OR)

La formule : //c**[\*[not(self::c)]//nom de balise etc. permet de préciser que la balise en question est bien dans le composant et non dans les composants enfants du composant**