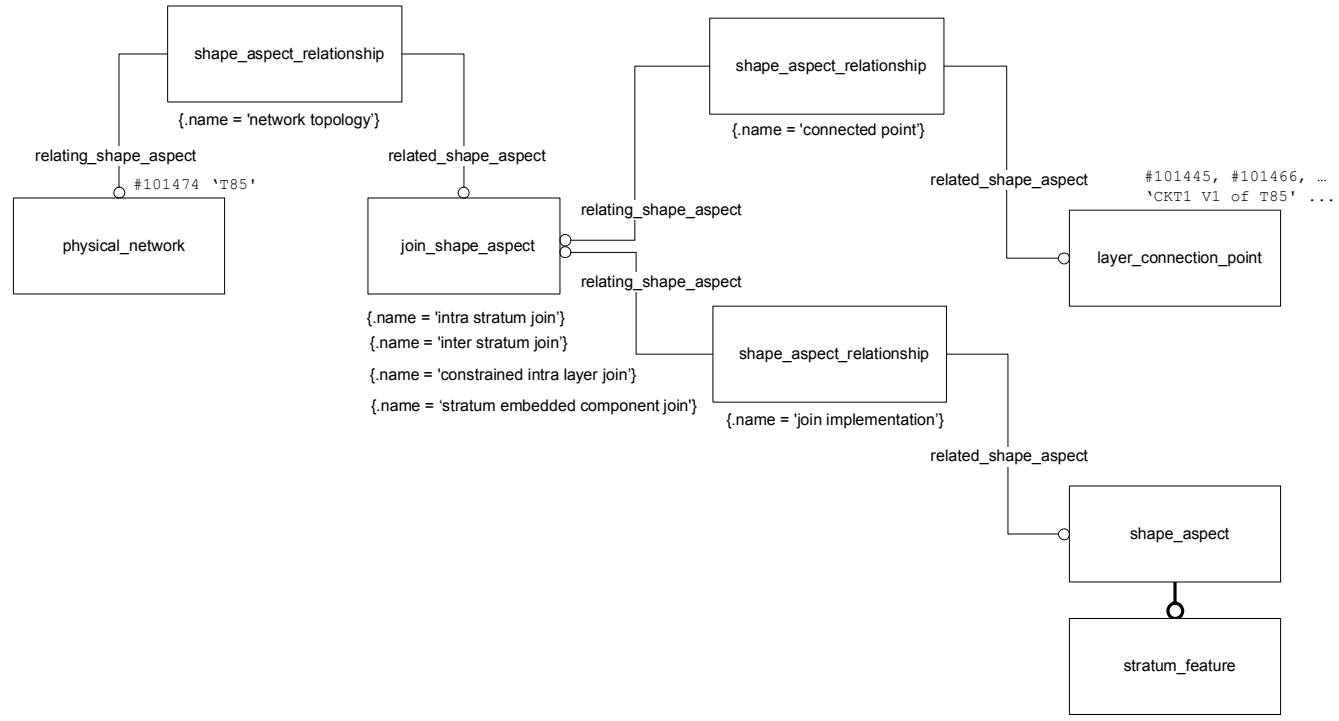
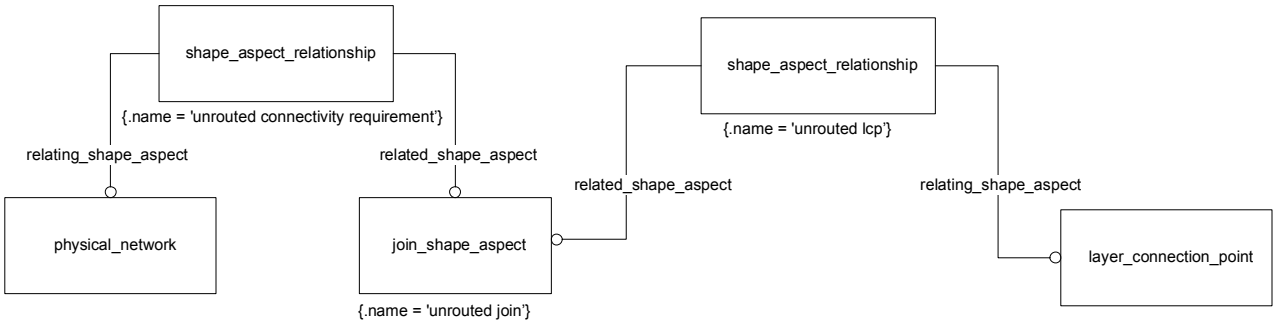
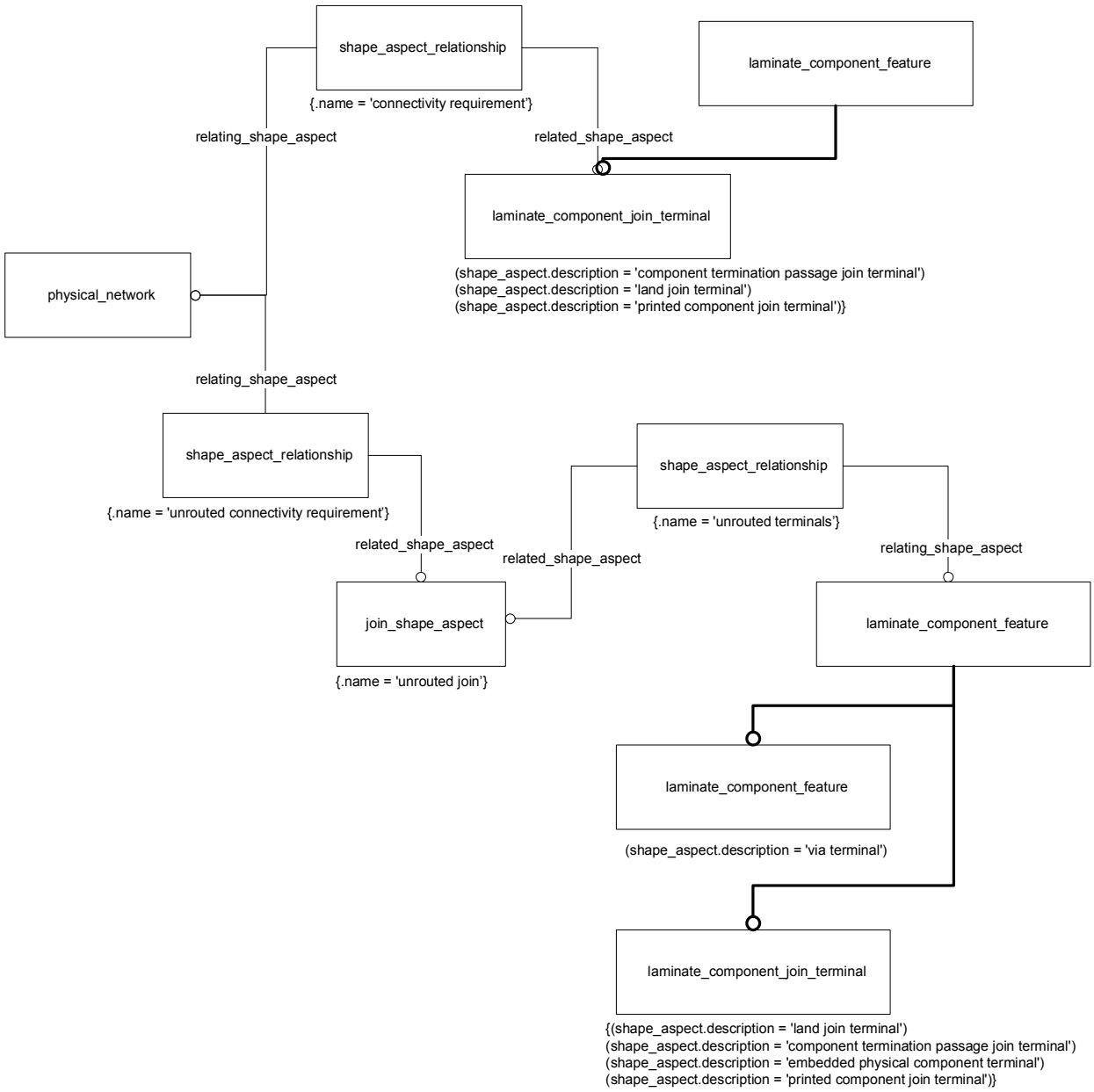
getLCPsForPhysicalNetwork	*Returns an aggregate of layer_connection_point containing all LCPs explicity joined to a physical_network.*
getLCFsForPhysicalNetwork	*Returns an aggregate of layer_connection_point containing all LCPs explicity joined to a physical_network. Query may be applied to either routed or unrouted physical networks.*
getInterStratumFeatureForDLLCP	*Returns an inter_stratum_feature associated with a dependently located layer_connection_point if one exists.*
getPPDLandsforDLLCP	Returns an aggregate of plated_passage_dependent_land associated with a 'dependently located' layer_connection_point if a plated_passage is the 'associated design object' of the layer_connection_point
getCSDLandsforDLLCP	Returns a contact_size_dependent_land associated with a 'dependently located' layer_connection_point if an interconnect_module_interface_terminal is the 'associated design object' of the DLLCP.
getCIEforLCP	Returns an aggregate of conductive_interconnect_element associated with a layer_connection_point if a 'conductive interconnect element terminal' is associated with the LCP.
getConnectedAreaComponentforLCP	Returns a connected_area_component that is associated with a given layer_connection_point if one exists.

getLCPsForPhysicalNetwork

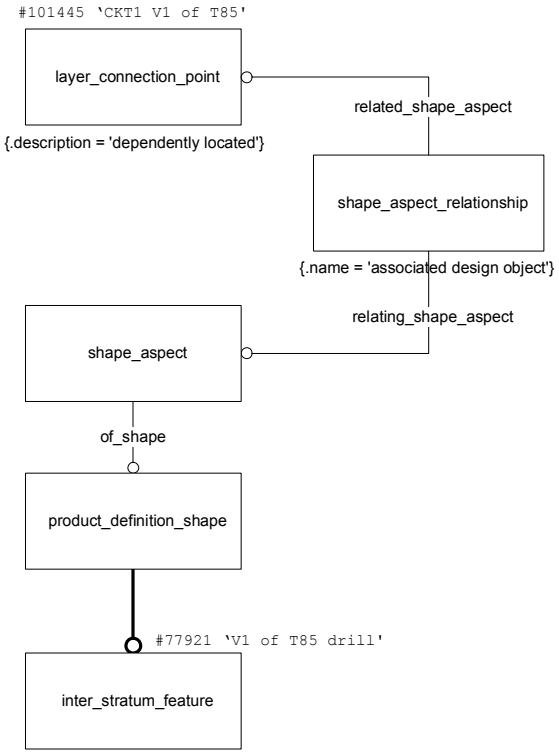




```
// Returns an aggregate of layer_connection_point containing all LCPs explicity joined to a physical_network.
// Query may be applied to either routed or unrouted physical networks.
// Uniqueness of the layer_connection_points contained in the aggregate is ensured by the implementation.

Aggregate<layer_connection_point> getLCPsForPhysicalNetwork(physical_network pn)
{
    Set<layer_connection_point> a_lcp = null;

    Aggregate <join_shape_aspect> a_jsa = relatedEntitiesOp(pn)
        where  {join_shape_aspect jsa}
                {shape_aspect_relationship sar}
                {pn <- sar.relating_shape_aspect}
                {sar.related_shape_aspect -> jsa}
                {sar.name='network topology'}

    if (SizeOf(a_jsa) >0)
    {
        For Each join_shape_aspect jsa Of a_jsa
        {
            Aggregate<layer_connection_point> a_lcpi = relatedEntitiesOp(jsa)
                where  {layer_connection_point lcp}
                        {shape_aspect_relationship sar}
                        {jsa <- sar.relating_shape_aspect}
                        {sar.related_shape_aspect -> lcp}
                        {sar.name='connected point'}

            For Each Layer_connection_point lcp Of a_lcpi
                Add lcp to a_lcp
        }

    return a_lcp
}
```

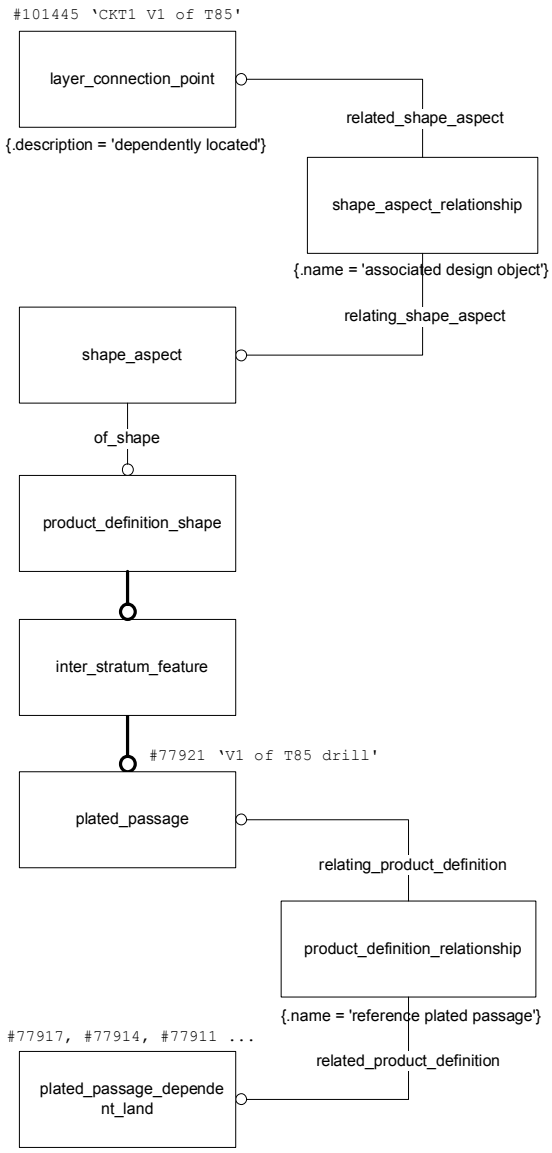

shape_aspect_relationship
{.name = 'connectivity requirement'}
laminate_component_feature
relating_shape_aspect
related_shape_aspect
laminate_component_join_terminal
(shape_aspect.description = 'component termination passage join terminal')
(shape_aspect.description = 'land join terminal')
(shape_aspect.description = 'printed component join terminal')}
physical_network
relating_shape_aspect
shape_aspect_relationship
{.name = 'unrouted connectivity requirement'}
shape_aspect_relationship
{.name = 'unrouted terminals'}
related_shape_aspect
related_shape_aspect
relating_shape_aspect
join_shape_aspect
{.name = 'unrouted join'}
laminate_component_feature
laminate_component_feature
(shape_aspect.description = 'via terminal')
laminate_component_join_terminal
{(shape_aspect.description = 'land join terminal')
(shape_aspect.description = 'component termination passage join terminal')
(shape_aspect.description = 'embedded physical component terminal')
(shape_aspect.description = 'printed component join terminal')}

```
// Returns an aggregate of laminate_component_feature containing the LCFs explicity joined to a physical_network.
// Query may be applied to either routed or unrouted physical networks.
// Uniqueness of the laminate_component_feature contained in the aggregate is ensured by the implementation.

Aggregate<Laminate_component_feature> getLCFsForPhysicalNetwork(physical_network e_pn)
    {
        Aggregate<Laminate_component_feature> a_lcf_for_pn = new Aggregate<Laminate_component_feature>
        Set<Laminate_component_feature> Set_lcf_for_pn = Set<Laminate_component_feature>

        Aggregate<Laminate_component_feature> a_lcf_cr = relatedEntitiesOp(e_pn)
            where  {laminate_component_feature e_lcf}
                   {shape_aspect_relationship e_sar}
                   {e_pn <- e_sar.relating_shape_aspect }
                   {e_sar.name = 'connectivity requirement'}
                   {e_sar.related_shape_aspect -> e_lcf}

        For each laminate_component_feature e_lcf_cr in a_lcf_cr
        {
            if (not (Set_lcf_for_pn.contains(e_lcf)))
            {
                add e_lcf to a_lcf_for_pn
                add e_lcf to Set_lcf_for_pn
            }
        }


        Aggregate<join_shape_aspect> a_jsa = relatedEntitiesOp(e_pn)
            where  {join_shape_aspect e_jsa}
                   {shape_aspect_relationship e_sar}
                   {e_pn <- e_sar.relating_shape_aspect }
                   {e_sar.name = 'unrouted connectivity requirement'}
                   {e_sar.related_shape_aspect -> e_jsa}

        For each join_shape_aspect e_jsa in a_jsa
        {
            Aggregate<Laminate_component_feature> a_lcfi = relatedEntitiesOp(e_jsa)
            where  {laminate_component_feature e_lcf}
                   {shape_aspect_relationship e_sar}
                   {e_jsa <- e_sar.related_shape_aspect }
                   {e_sar.name = 'unrouted terminals'}
                   {e_sar.relating_shape_aspect -> e_lcf}

            For each laminate_component_feature e_lcf in a_lcfi
            {
                if (not(Set_lcf_for_pn.contains(e_lcf)))
                {
                    add e_lcf to a_lcf_for_pn
                    add e_lcf to Set_lcf_for_pn
                }
            }
        }
        return a_lcf_for_pn;
    }
```

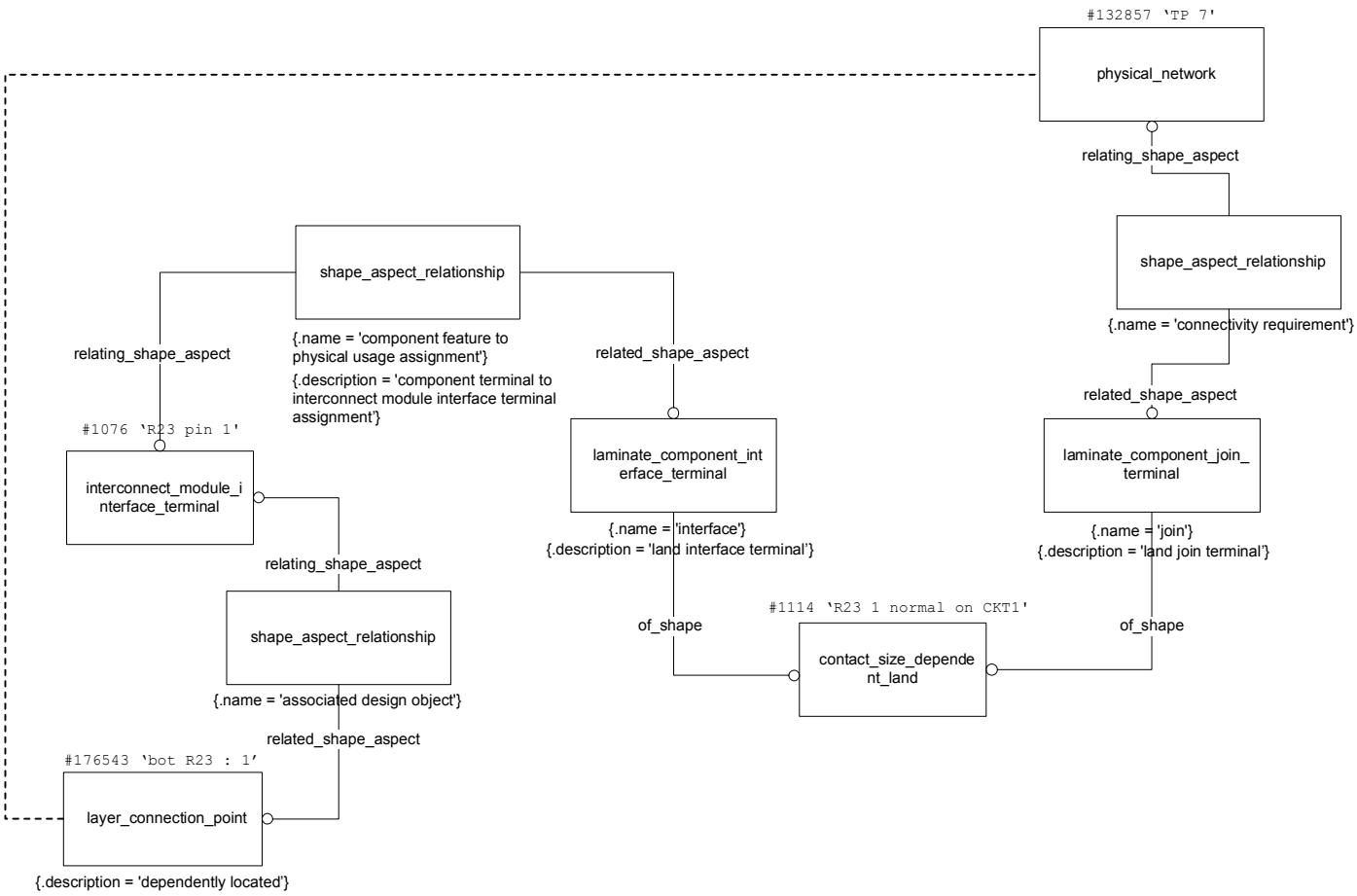


// Returns an inter_stratum_feature associated with a dependently located layer_connection_point if one exists.

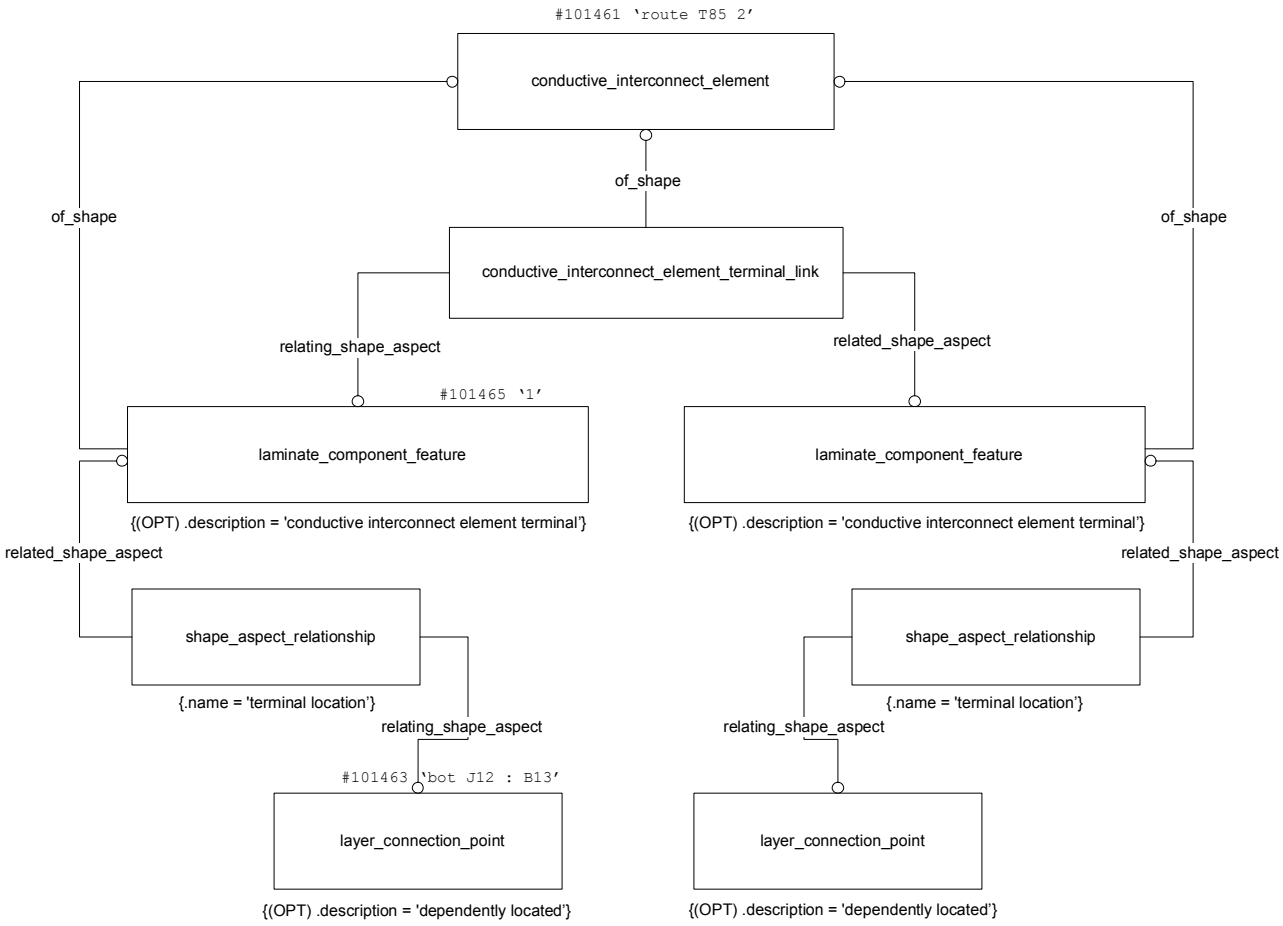
```
inter_stratum_feature  getInterStratumFeatureForDLLCP(layer_connection_point lcp)
{
    If not(layer_connection_point.description == 'dependently located')
        return null

    shape_aspect sa = relatedEntityOp(lcp)
        where  {shape_aspect_relationship sar}
                {lcp <- sar.related_shape_aspect}
                {sar.relating_shape_aspect -> sa}
                {sar.name='associated design object'}

    inter_stratum_feature isf =  referencedEntityOp(sa)
        where  {sa.of_shape -> isf}

    return isf
}
```



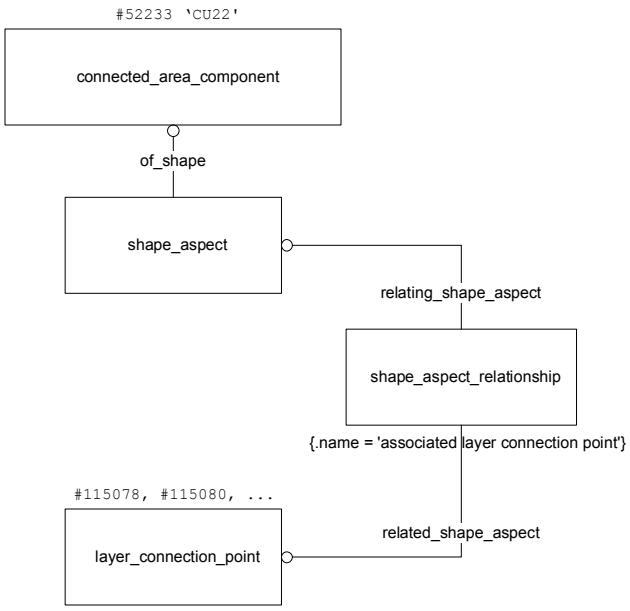

```
// Returns an aggregate of plated_passage_dependent_land associated with a 'dependently located' layer_connection_point if a
// plated_passage is the 'associated design object' of the layer_connection_point

Aggregate<plated_passage_dependent_land> getPPDLandsforDLLCP(layer_connection_point lcp)
{
    inter_stratum_feature isf = getInterStratumFeatureForDLLCP(lcp)

    if not (isf InstanceOf plated_passage)
        return null

    Aggregate<plated_passage_dependent_land> a_ppdl = relatedEntitiesOp(inter_stratum_feature isf)
        where {plated_passage_dependent_land ppdl}
                {product_definition_relationship pdr}
                {isf <- pdr.relating_product_definition}
                {pdr.related_product_definition -> ppdl}
                {pdr.name='reference plated passage'}

    return a_ppdl
}
```




```
// Returns a contact_size_dependent_land associated with a 'dependently located' layer_connection_point if an
// interconnect_module_interface_terminal is the 'associated design object' of the DLLCP

contact_size_dependent_land getCSDLandforDLLCP(layer_connection_point lcp)
{
    If not(layer_connection_point.description == 'dependently located')
        return null

    interconnect_module_interface_terminal imit = relatedEntityOp(lcp)
        where   {shape_aspect_relationship sar}
                {lcp <- sar.related_shape_aspect}
                {sar.relating_shape_aspect -> imit}
                {sar.name='associated design object'}

    if (imit == null)
        return null

    laminate_component_interface_terminal lcit = relatedEntityOp(imit)
        where   {shape_aspect_relationship sar}
                {imit <- sar.relating_shape_aspect}
                {sar.related_shape_aspect -> lcit}
                {sar.description='component terminal to interconnect module interface terminal assignment'}

    if (lcit == null)
        return null

    if not(lcit.description == 'land interface terminal')
        return null

    contact_size_dependent_land csdl = referencedEntityOp(lcit)
            where   {lcit.of_shape->csdl}

    return csdl
    }
```




```
// Returns an aggregate of conductive_interconnect_element associated with a layer_connection_point if a
// 'conductive interconnect element terminal' is associated with the LCP

Aggregate<conductive_interconnect_element> getCIEforLCP(layer_connection_point lcp)
{
    Aggregate<conductive_interconnect_element> a_cie

    Aggregate<laminate_component_feature> a_lcf = relatedEntitiesOp(lcp)
        where  {laminate_component_feature lcf}
                {shape_aspect_relationship sar}
                {lcp <- sar.relating_shape_aspect}
                {sar.related_shape_aspect -> lcf}
                {sar.name='terminal location'}

    For Each laminate_component_feature lcf in a_lcf
    {
        If (lcf.description == 'conductive interconnect element terminal')
        {
            conductive_interconnect_element cie = referencedEntityOp(lcf)
                where  {lcf.of_shape->cie}
            Add unique cie to a_cie
        }
    }

    return a_cie
}
```



// Returns a connected_area_component that is associated with a given layer_connection_point if one exists.

```
connected_area_component getConnectedAreaComponentforLCP(layer_connection_point lcp)
{
    shape_aspect sa = relatedEntityOp(lcp)
        where {shape_aspect_relationship sar}
                {lcp <- sar.related_shape_aspect}
                {sar.relating_shape_aspect -> sa}
                {sar.name='associated layer connection point'}

    If (sa == null)
        return null

    connected_area_component cac = referencedEntityOp(sa)
        where {sa.of_shape->cac}

    return cac
}
```