```
/*
#################################################################################
#########################
# Rocket League (220224.66435.3685966/5/2024) SDK
# Generated with the UE3SDKGenerator v2.2.7
#
#################################################################################
########## #
# File: Core_classes.hpp
#
#################################################################################
########## #
# Credits: TheFeckless, ItsBranK
# Links: www.github.com/itsbrank/UE3SDKGenerator, www.twitter.com/itsbrank
#################################################################################
#########################
*/
#pragma once

#ifdef _MSC_VER
#pragma pack(push, 0x8)
#endif

/*
#
#################################################################################
########## #
# Constants
#
#################################################################################
########## #
*/

#define CONST_ZeroRotator                    Rot(0,0,0)
#define CONST_ZeroVector                     Vect(0,0,0)
#define CONST_UpVector                       Vect(0,0,1)
#define CONST_RightVector                    Vect(0,1,0)
#define CONST_ForwardVector                  Vect(1,0,0)
#define CONST_InvAspectRatio16x9               0.56249
#define CONST_InvAspectRatio5x4                0.8
#define CONST_InvAspectRatio4x3                0.75
#define CONST_AspectRatio16x9                  1.77778
#define CONST_AspectRatio5x4                   1.25
#define CONST_AspectRatio4x3                   1.33333
#define CONST_INDEX_NONE                       -1
#define CONST_UnrRotToDeg                      0.00549316540360483
#define CONST_DegToUnrRot                      182.0444
#define CONST_RadToUnrRot                      10430.3783504704527
#define CONST_UnrRotToRad                      0.00009587379924285
#define CONST_DegToRad                       0.017453292519943296
#define CONST_RadToDeg                       57.295779513082321600
#define CONST_Pi                          3.1415926535897932
#define CONST_MaxQWORD                         0xFFFFFFFFFFFFFFFF
#define CONST_MinInt                        0x80000000
```

```
#define CONST_MaxInt                    0x7fffffff

/*
#
# ============================================================================
========== #
# Enums
#
# ============================================================================
========== #
*/

// Enum Core.Object.EAspectRatioAxisConstraint
enum class EAspectRatioAxisConstraint : uint8_t
{
AspectRatio_MaintainYFOV                = 0,
AspectRatio_MaintainXFOV                = 1,
AspectRatio_MajorAxisFOV                = 2,
AspectRatio_END                         = 3
};

// Enum Core.Object.EDebugBreakType
enum class EDebugBreakType : uint8_t
{
DEBUGGER_NativeOnly                     = 0,
DEBUGGER_ScriptOnly                     = 1,
DEBUGGER_Both                           = 2,
DEBUGGER_END                            = 3
};

// Enum Core.Object.EAutomatedRunResult
enum class EAutomatedRunResult : uint8_t
{
ARR_Unknown                             = 0,
ARR_OOM                                 = 1,
ARR_Passed                              = 2,
ARR_END                                 = 3
};

// Enum Core.Object.EInterpCurveMode
enum class EInterpCurveMode : uint8_t
{
CIM_Linear                              = 0,
CIM_CurveAuto                           = 1,
CIM_Constant                            = 2,
CIM_CurveUser                           = 3,
CIM_CurveBreak                          = 4,
CIM_CurveAutoClamped                    = 5,
CIM_END                                 = 6
};

// Enum Core.Object.EInterpMethodType
enum class EInterpMethodType : uint8_t
{
```

```
IMT_UseFixedTangentEvalAndNewAutoTangents = 0,
IMT_UseFixedTangentEval = 1,
IMT_UseBrokenTangentEval = 2,
IMT_END = 3
};

// Enum Core.Object.EAxis
enum class EAxis : uint8_t
{
AXIS_NONE = 0,
AXIS_X = 1,
AXIS_Y = 2,
AXIS_BLANK = 3,
AXIS_Z = 4,
AXIS_END = 5
};

// Enum Core.Object.ETickingGroup
enum class ETickingGroup : uint8_t
{
TG_PreAsyncWork = 0,
TG_DuringAsyncWork = 1,
TG_PostAsyncWork = 2,
TG_PostUpdateWork = 3,
TG_EffectsUpdateWork = 4,
TG_END = 5
};

// Enum Core.Object.EInputEvent
enum class EInputEvent : uint8_t
{
IE_Pressed = 0,
IE_Released = 1,
IE_Repeat = 2,
IE_DoubleClick = 3,
IE_Axis = 4,
IE_END = 5
};

// Enum Core.Object.AlphaBlendType
enum class EAlphaBlendType : uint8_t
{
ABT_Linear = 0,
ABT_Cubic = 1,
ABT_Sinusoidal = 2,
ABT_EaseInOutExponent2 = 3,
ABT_EaseInOutExponent3 = 4,
ABT_EaseInOutExponent4 = 5,
ABT_EaseInOutExponent5 = 6,
ABT_END = 7
};

// Enum Core._Types_Core.OnlinePlatform
enum class EOnlinePlatform : uint8_t
```

```
{
OnlinePlatform_Unknown                  = 0,
OnlinePlatform_Steam                    = 1,
OnlinePlatform_PS4                      = 2,
OnlinePlatform_PS3                      = 3,
OnlinePlatform_Dingo                    = 4,
OnlinePlatform_QQ_DEPRECATED            = 5,
OnlinePlatform_OldNNX                   = 6,
OnlinePlatform_NNX                      = 7,
OnlinePlatform_PsyNet                   = 8,
OnlinePlatform_Deleted                  = 9,
OnlinePlatform_WeGame_DEPRECATED        = 10,
OnlinePlatform_Epic                     = 11,
OnlinePlatform_END                      = 12
};

// Enum Core._Types_Core.EImageType
enum class EImageType : uint8_t
{
EIT_Unknown                     = 0,
EIT_JPEG                        = 1,
EIT_PNG                         = 2,
EIT_END                         = 3
};

// Enum Core._Types_Core.EInputAPI
enum class EInputAPI : uint8_t
{
InputAPI_Default                = 0,
InputAPI_SteamInput             = 1,
InputAPI_END                    = 2
};

// Enum Core._Types_Core.EFlushResult
enum class EFlushResult : uint8_t
{
FlushResult_Success             = 0,
FlushResult_InProgress          = 1,
FlushResult_TimedOut            = 2,
FlushResult_END                 = 3
};

// Enum Core._Types_Core.EVoiceResultCode
enum class EVoiceResultCode : uint8_t
{
VRC_Success                     = 0,
VRC_NoConnection                = 1,
VRC_InvalidCredentials          = 2,
VRC_TooManyParticipants         = 3,
VRC_UserKicked                  = 4,
VRC_UserBanned                  = 5,
VRC_ServiceFailure              = 6,
VRC_AccessDenied                = 7,
VRC_UnexpectedError             = 8,
```

```
VRC_END                                 = 9
};

//// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate29
//enum class EContentKeyIndex_PrimeUpdate29 : uint8_t
//{
//    PrimeUpdate29_AE206DA0E0A3AAD8B6755870B27FA65E     = 0,
//    PrimeUpdate29_30CA52092D2CEDAA55E764986CC47D60     = 1,
//    PrimeUpdate29_350E4C7D232183351C9A7CA19AF1D171     = 2,
//    PrimeUpdate29_0F4D063A56589D60F7C667659284EF79     = 3,
//    PrimeUpdate29_DF40E515A9C2BC9F9B6459DA06BA12B4     = 4,
//    PrimeUpdate29_07DAE5D877865DA3A7B27525C6BCD772     = 5,
//    PrimeUpdate29_398C669A96F0D8C7A7CB85C6B7F9D40D     = 6,
//    PrimeUpdate29_1A0A172C108D12F6B9E2582B8353515A     = 7,
//    PrimeUpdate29_DFA1AA4962EC1FAF6388A5D29978701F     = 8,
//    PrimeUpdate29_82BD9EB94679B73DA8574C6DB2C8737E     = 9,
//    PrimeUpdate29_6E59D0C3AAAF6C9D2E326F4293999A64     = 10,
//    PrimeUpdate29_3F53F796EEBEDA376ACA7199F20CAA63     = 11,
//    PrimeUpdate29_C29913CE0063B6A4499E4AFF4C5D56D1     = 12,
//    PrimeUpdate29_88F2B75897AE1A5B80B4DCC82C376200     = 13,
//    PrimeUpdate29_EF28529F54D54075C40BA9309352A504     = 14,
//    PrimeUpdate29_4F3D2425A947760B6BD53B2E51290AE9     = 15,
//    PrimeUpdate29_42DC275ACD6AF3B20C30E2A016AECC19     = 16,
//    PrimeUpdate29_ADC297BC151083F6529E1A1559605351     = 17,
//    PrimeUpdate29_755715CBB92570E53F72C1BEF66C2E93     = 18,
//    PrimeUpdate29_907AF3C9F3A0262587FF66ADF42F4D3F     = 19,
//    PrimeUpdate29_C271DB65745C937C51A4591C915C5AF0     = 20,
//    PrimeUpdate29_END                                  = 21
//};
//
//
// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate30
//enum class EContentKeyIndex_PrimeUpdate30 : uint8_t
//{
//    PrimeUpdate30_1DB64BB9636815EFA9399659109DD68D     = 0,
//    PrimeUpdate30_9234CF28DDEC42BA6D348267FA446B0A     = 1,
//    PrimeUpdate30_503AD462D2DCE768AE47BB2329144CAE     = 2,
//    PrimeUpdate30_213CA336DDB319FD27DEA933E3501CB2     = 3,
//    PrimeUpdate30_F7FD01F704F37BC24F42A1F9316D8A1D     = 4,
//    PrimeUpdate30_3087684ADB9F22472E3AFD9BE5BF94F5     = 5,
//    PrimeUpdate30_810B945ECFD25E0822A159B328604ECA     = 6,
//    PrimeUpdate30_5465981D30DE3D6E327B176D436FEF9D     = 7,
//    PrimeUpdate30_9A9242C1EC822F5CABB949FFDF01D97A     = 8,
//    PrimeUpdate30_452F33B1387640B277D05CCF28B27094     = 9,
//    PrimeUpdate30_33430E4B5498C30E9942BC67BB35E032     = 10,
//    PrimeUpdate30_712FAB1F622E61A50F98B76312770661     = 11,
//    PrimeUpdate30_2386DB5AA955DDFF74A6A7AF443249B5     = 12,
//    PrimeUpdate30_C5BE8F988BD1FF6A53893EC1B454B272     = 13,
//    PrimeUpdate30_ED667588F3F916C76D11EADB27036255     = 14,
//    PrimeUpdate30_27C7319E6A9E8651E89204245770107E     = 15,
//    PrimeUpdate30_966DAC7787B441D0E6195D90634ADFD1     = 16,
//    PrimeUpdate30_DF823FCBBF433C11264736998336CFB9     = 17,
//    PrimeUpdate30_DC5ABAEFCD0A266C64BF8664FEE15309     = 18,
//    PrimeUpdate30_22EC7532DB37341C045127F6263A15FF     = 19,
```

```
//    PrimeUpdate30_28B8BBA8CBF5B1B73BD72127D5ADCCB5    = 20,
//    PrimeUpdate30_AA915328B1E7A7251488C75194A90384    = 21,
//    PrimeUpdate30_D76E2A921047B235911605B9B008F606    = 22,
//    PrimeUpdate30_55026AC8526C9B85A556C370FFCBE521    = 23,
//    PrimeUpdate30_84C1F1262EF1E5DC120A7ED88DBFEF15    = 24,
//    PrimeUpdate30_A99C4C15B5858BB823253B065BE66BB2    = 25,
//    PrimeUpdate30_END                                 = 26
//};
//
////// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate31
//enum class EContentKeyIndex_PrimeUpdate31 : uint8_t
//{
//    PrimeUpdate31_8416552FC13C775C6325D95DAFD2467C    = 0,
//    PrimeUpdate31_4D7DD3DF13D1A7A874987ED3471E73C8    = 1,
//    PrimeUpdate31_880C8C1A2B7E0D89E198FD070B76C948    = 2,
//    PrimeUpdate31_85F5966A74411E8888F7228B41775B64    = 3,
//    PrimeUpdate31_0F3CBA7F9849D469C134A6FF10453DB2    = 4,
//    PrimeUpdate31_F9E087849D5F5873B1D9C2C2A76E8280    = 5,
//    PrimeUpdate31_4F3016BFEEF5686ABF5BCC01EDAB34AB    = 6,
//    PrimeUpdate31_289E688E31D67B07097C3607B0E4B766    = 7,
//    PrimeUpdate31_FDFE789EC592F2AE370CBDC6644331EA    = 8,
//    PrimeUpdate31_3B62BABAB19A41542A3F1A543BA902C7    = 9,
//    PrimeUpdate31_2ED307705C547AAE42F52929ED345B54    = 10,
//    PrimeUpdate31_88C241C0F02B450B51F0A9AF5DDC359D    = 11,
//    PrimeUpdate31_94713368AC068D293F842AD501456252    = 12,
//    PrimeUpdate31_7EDA3FFCCA3D799DD992CB9E6E1641BA    = 13,
//    PrimeUpdate31_597E29169FCA2B5E0022CB3C17FD6276    = 14,
//    PrimeUpdate31_B403B6BC53473983E71404459C5C329D    = 15,
//    PrimeUpdate31_DA7EA18B2A78CD2DC80B1647AC96CB4D    = 16,
//    PrimeUpdate31_3CE5ABD97423D57980D4CE4984D23723    = 17,
//    PrimeUpdate31_END                                 = 18
//};
//
////// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate32
//enum class EContentKeyIndex_PrimeUpdate32 : uint8_t
//{
//    PrimeUpdate32_300CBBF87113F4C1E17912EAAEACCF62    = 0,
//    PrimeUpdate32_AC64316796AD89170208EA1ADE72EB53    = 1,
//    PrimeUpdate32_9F4361589D74380E0351FCF10A7F2032    = 2,
//    PrimeUpdate32_950D6028A559ADDDA77671B90D707A89    = 3,
//    PrimeUpdate32_BBCA9FD0992D35DE029D78D34001A3A8    = 4,
//    PrimeUpdate32_1BAB656F9A701BF05B8483F5C3E95365    = 5,
//    PrimeUpdate32_EB99CF7F2BD77FB03F91DB208B6E3E78    = 6,
//    PrimeUpdate32_D1993DE64E44D83188D68A0052953321    = 7,
//    PrimeUpdate32_43520ABD8E211AE3EFA707DE4736D7A6    = 8,
//    PrimeUpdate32_595CDCCADBF7CF6359DCD9CACA14BF45    = 9,
//    PrimeUpdate32_EFEB90A98D01EF42CACDF3C6166A4E6B    = 10,
//    PrimeUpdate32_B87BF5D0B9F7DE0B688739F591EEC514    = 11,
//    PrimeUpdate32_7AB8B7644C5D85D344D09FF1468C412C    = 12,
//    PrimeUpdate32_45F3ADEBE4EB02ABAF7B13185A766207    = 13,
//    PrimeUpdate32_1427133C46AA4BAD6F54BC030AC7F9A9    = 14,
//    PrimeUpdate32_61C0088685FFBE4E0D6F438B118E5987    = 15,
//    PrimeUpdate32_84190309583ABA308C0C2E6621E37B9E    = 16,
//    PrimeUpdate32_DE9085DF47CD080E0BC98052B543E1CA    = 17,
```

```
//    PrimeUpdate32_D71B3C1F2C6CCAE81795D360C9E51B87     = 18,
//    PrimeUpdate32_FE659BEDDCA2DCBEBFD0D5A71CE5F55E     = 19,
//    PrimeUpdate32_C4CB8C293E93DB50EC6C29DE36C52B90     = 20,
//    PrimeUpdate32_CE0E00A1192764DEF4798337C1ACE048     = 21,
//    PrimeUpdate32_D5BAADF84D6B7ACAC93AC22539969043     = 22,
//    PrimeUpdate32_70F2913F8A944F8E9F578CE3F8D789B2     = 23,
//    PrimeUpdate32_374EBD593462DC7180BD4F2F8785F548     = 24,
//    PrimeUpdate32_851BE21BC1A5E102E86B4239FF2C7645     = 25,
//    PrimeUpdate32_E32780FE596DDD89FDA1CED46D30BA08     = 26,
//    PrimeUpdate32_1C213CD84626E67147C1AAF044BCE949     = 27,
//    PrimeUpdate32_AE65305403939F84C0F5DE5775770C7A     = 28,
//    PrimeUpdate32_0EE4F6E8266F4BAA55F56AA6CAED927E     = 29,
//    PrimeUpdate32_654A14F505D929128335F21A88B72936     = 30,
//    PrimeUpdate32_B80CDD1F4B9BE4A5C31CF96645FDFAD6     = 31,
//    PrimeUpdate32_CA43F5822A576ED8067999E2A43C82FC     = 32,
//    PrimeUpdate32_C5C0A837BAC4698316A50EA505F4345D     = 33,
//    PrimeUpdate32_F082F05E7D651EDE256CCCC9A0699E15     = 34,
//    PrimeUpdate32_D845ECFEEB577D15E6204540A327CD8A     = 35,
//    PrimeUpdate32_9B27CA53E1CD25849873E3B0DEAA4265     = 36,
//    PrimeUpdate32_66EE45220A851279ECFA9ACDBB520988     = 37,
//    PrimeUpdate32_END                                  = 38
//};
//
//
// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate33
//enum class EContentKeyIndex_PrimeUpdate33 : uint8_t
//{
//    PrimeUpdate33_9859F4962ED9225261C1735A9B0F6A7F     = 0,
//    PrimeUpdate33_C212DFEF60853EEB61ACA8181A89A15F     = 1,
//    PrimeUpdate33_C418C87964E1BB0C7DCB3E70779FB44B     = 2,
//    PrimeUpdate33_1FEB11B693427C63E457203E461639E3     = 3,
//    PrimeUpdate33_4ED08FC09FB52F44407423877E04CACA     = 4,
//    PrimeUpdate33_DBFAF049AEB30397815B08553C60FB59     = 5,
//    PrimeUpdate33_8A88FA4B1CFBE1E21536C648BBD4F23E     = 6,
//    PrimeUpdate33_F59B6D4EFACF226D4B590EB3B22A9C34     = 7,
//    PrimeUpdate33_B435739A7F1C97EF5F1D7554AC117749     = 8,
//    PrimeUpdate33_3920915610E4452D9178C330AC20EBFA     = 9,
//    PrimeUpdate33_BE72EF04B58AF461A7D85A77B48EE37A     = 10,
//    PrimeUpdate33_E7BE7D7122884CEEF63FBC297632C761     = 11,
//    PrimeUpdate33_F23A298C56E7EBF4CF8C6875F03BFE72     = 12,
//    PrimeUpdate33_ECB007102C74731082C7272DD98D502A     = 13,
//    PrimeUpdate33_618F822EFD2AA0D7B69A35C84C600564     = 14,
//    PrimeUpdate33_E80A98F4841B3861D085BBE347233903     = 15,
//    PrimeUpdate33_5EE9F2BC8C9CA242782C40CF779804CE     = 16,
//    PrimeUpdate33_3A7BDFD41C45E477C990F83772DDE5FF     = 17,
//    PrimeUpdate33_ED459D7A6FE5DBCECABA1F93BD49247B     = 18,
//    PrimeUpdate33_6C677BD35A184A2BB0EBAB421BC55E39     = 19,
//    PrimeUpdate33_7ED0D8CF98C35C1549F175290231FC02     = 20,
//    PrimeUpdate33_E48D95A331BB274C1F99297A1AEF0A9A     = 21,
//    PrimeUpdate33_78C0060D1C5191F62C68D4DED6FCE798     = 22,
//    PrimeUpdate33_15ABD202609BD1609F930B708BCD7208     = 23,
//    PrimeUpdate33_3F18B0F70069D6ADF8C0E2B783B36BC5     = 24,
//    PrimeUpdate33_9AF0D8EDFCB91756D4F7048012E922D6     = 25,
//    PrimeUpdate33_BE2A12592777B870F55FAD163C043192     = 26,
```

```
//    PrimeUpdate33_EAA00740DF8EF37816AE479E8CCB20B6     = 27,
//    PrimeUpdate33_7110D96C5620E2A3360E3887D254A2D6     = 28,
//    PrimeUpdate33_0F7338535499E16C2BF19CB546C7C2A6     = 29,
//    PrimeUpdate33_E168E8341CB79082D0D9AA4D1FEC97A8     = 30,
//    PrimeUpdate33_5C1562652DB849A2BB8B1779333E573D     = 31,
//    PrimeUpdate33_2E512914EDDC939A207DE0F77DDA26CF     = 32,
//    PrimeUpdate33_CEE13EA70861968B6AAA07CFA66162DB     = 33,
//    PrimeUpdate33_END                                  = 34
//};
//
//
// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate34
//enum class EContentKeyIndex_PrimeUpdate34 : uint8_t
//{
//    PrimeUpdate34_92E48285E52D9C5376465AFA3C483D21     = 0,
//    PrimeUpdate34_798E5021EB2EE0D9150525103009597E     = 1,
//    PrimeUpdate34_3E57F996CC259382D4F9D60DD3411999     = 2,
//    PrimeUpdate34_980D75703713C7B361FA6663369415DD     = 3,
//    PrimeUpdate34_0BB7B0C3C62B7BC62A8CF5AE4224313F     = 4,
//    PrimeUpdate34_16684068F60C3BEF59FCA6681DEFC634     = 5,
//    PrimeUpdate34_8638D056E206619C71F7358A3112AA75     = 6,
//    PrimeUpdate34_119863391AC5E311B66BF5EB49A7E392     = 7,
//    PrimeUpdate34_DCA6627ACE5BE1A3D82E699E38B383BA     = 8,
//    PrimeUpdate34_F44577CDD9D45CBF49CEDC32EE829FC1     = 9,
//    PrimeUpdate34_4D75D62B40DE98B190090EB2851E7598     = 10,
//    PrimeUpdate34_D0378645638D7FE611F959E71B989E88     = 11,
//    PrimeUpdate34_F67C94735DF24D72CE6E0983445EFF94     = 12,
//    PrimeUpdate34_677E188D9A579ADE3F2CD3747D225C24     = 13,
//    PrimeUpdate34_F7E58DABA865EC58A2D4496DE3C4375F     = 14,
//    PrimeUpdate34_C2BD319DE1987E7D4DD851CCA2A06FCC     = 15,
//    PrimeUpdate34_B8CBE903CC7AE40A7F668F6427E57098     = 16,
//    PrimeUpdate34_D0A8BEDB5EB633D376B288A6729F2958     = 17,
//    PrimeUpdate34_6D8E086B4A152E4BD9059DC67E9A5330     = 18,
//    PrimeUpdate34_D93828EBD836F1BEEF0BA345FF2D7D2B     = 19,
//    PrimeUpdate34_584D6260FEA4A45FBEC3729A47F32A2C     = 20,
//    PrimeUpdate34_525D1944EF49B6263813E9C33FE9A2E7     = 21,
//    PrimeUpdate34_EB7F8E2F3CD896BECB59816B49A8225D     = 22,
//    PrimeUpdate34_F53E63F89C250939560481ACF2EB4F57     = 23,
//    PrimeUpdate34_1EFD13089FD99C03404F686ABB842FC1     = 24,
//    PrimeUpdate34_CE1C329C9C4A36B5347CDDFC34E6FE74     = 25,
//    PrimeUpdate34_AA31999372C5419377BC328A124FA260     = 26,
//    PrimeUpdate34_1FEC83F0FCACE770E7D580C38DBA0B7C     = 27,
//    PrimeUpdate34_380201BF1433CCFDC60B75AA5A1F27A0     = 28,
//    PrimeUpdate34_0B5FB027AD52B9D07E7C6610F51D049B     = 29,
//    PrimeUpdate34_AAF86940AE1401B7904BB3F1B7F0BF84     = 30,
//    PrimeUpdate34_A2EEFC632E25737A361937883AA2B9E3     = 31,
//    PrimeUpdate34_25DFE6D61E4B7BB4B8F48EA4C2893C30     = 32,
//    PrimeUpdate34_9A1D6F3529477B911753DA025ADE429F     = 33,
//    PrimeUpdate34_1CA56D0C508FF3D00C3347E82B5396F8     = 34,
//    PrimeUpdate34_E648CD9F3659F31EF46715B3FB73EF16     = 35,
//    PrimeUpdate34_36892DC2D16FE232BA96D6E8B9E5288E     = 36,
//    PrimeUpdate34_AED24E43D476A4470C99AABEE2C53673     = 37,
//    PrimeUpdate34_32B30CAF516C298774B59FC806B3B6AA     = 38,
//    PrimeUpdate34_02A902A86328CF57E63A7D1AD763DE60     = 39,
```

```
//    PrimeUpdate34_C4C41667DE1E754135DE7DB5710A24DD     = 40,
//    PrimeUpdate34_6AF47E33D558FC35C2FEB7A9D9ACAC59     = 41,
//    PrimeUpdate34_9C6ABEFD5E0C4F2DE3A2F7D146FAF952     = 42,
//    PrimeUpdate34_F3CA5A2918490945C363517319A0C2EC     = 43,
//    PrimeUpdate34_8592B9C622C8A3189DB3B4E2C2362142     = 44,
//    PrimeUpdate34_AC0C2D53149AD444AEFA1F9F7F29D39E     = 45,
//    PrimeUpdate34_074B3338DD2160921DB2DF191227F03E     = 46,
//    PrimeUpdate34_BF336D6E7DAC9A696838C3A788CEA62E     = 47,
//    PrimeUpdate34_67FB8392992984DA262E637A376E2318     = 48,
//    PrimeUpdate34_775260504F332F70DB926CD3CD7E63A7     = 49,
//    PrimeUpdate34_258F7CD5F76794C5180F23D0AB71E40C     = 50,
//    PrimeUpdate34_6177123148139D8930D4DFEA95501359     = 51,
//    PrimeUpdate34_A09893A74F7796FC0D23ADDF21E51A68     = 52,
//    PrimeUpdate34_72481FA8D257BD5935DC6BFAAFFE37B2     = 53,
//    PrimeUpdate34_A8A33573BBF97BC994426669BA945F97     = 54,
//    PrimeUpdate34_43A538B20CFD5AE55EBE8327D70618AA     = 55,
//    PrimeUpdate34_C309EEE625F26444CB456A7C3738519D     = 56,
//    PrimeUpdate34_40370551AA833BA691B954D5F223D20E     = 57,
//    PrimeUpdate34_1D49DD289EEBB9EF24BD0DC1DC8790C7     = 58,
//    PrimeUpdate34_AAE20A59233757624633935 6A56D13CC     = 59,
//    PrimeUpdate34_44C670308CD575930F330390E81F5DFD     = 60,
//    PrimeUpdate34_7EA963419DAF9F1F789AA0BCFA32504F     = 61,
//    PrimeUpdate34_418B733466B3543D3FF771D598D8E4F8     = 62,
//    PrimeUpdate34_185C6E063F04824C1408EA9670595259     = 63,
//    PrimeUpdate34_70F0016A0CCAAB0D71EDF0DA29C52171     = 64,
//    PrimeUpdate34_3F24725E47D67DD20FE3AD9F35AEF209     = 65,
//    PrimeUpdate34_END                                  = 66
//};
//
//
// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate35
//enum class EContentKeyIndex_PrimeUpdate35 : uint8_t
//{
//    PrimeUpdate35_904C03345B5306946918E821FFD0557D     = 0,
//    PrimeUpdate35_7D2BD3C4466DC9DCB376054A00FD8704     = 1,
//    PrimeUpdate35_95D345AF6F80E57FF039F0AF309CF397     = 2,
//    PrimeUpdate35_8895661B346552E41E98626288616FDB     = 3,
//    PrimeUpdate35_4936E2EC04B35A4C33132DB942EEFBF9     = 4,
//    PrimeUpdate35_E4CE5BA9AE7B7CF44E87C642A1B2075F     = 5,
//    PrimeUpdate35_AB8CA32A6AC22D1BCD733B45AAF3E516     = 6,
//    PrimeUpdate35_82504C8CB31FDBF83749FDB9D4647B22     = 7,
//    PrimeUpdate35_0F79BF2A330B5F14F6F1B334BAD190B8     = 8,
//    PrimeUpdate35_2FE48155423D40CAD4A7C1E13DFCA010     = 9,
//    PrimeUpdate35_85B7C89F12D13A7FC7C1BF63F2223D07     = 10,
//    PrimeUpdate35_8C08D9BB2ECFFE7FCB5E1C9ADA0C6915     = 11,
//    PrimeUpdate35_30D1E752D5CC3D535314F48E0155BC8B     = 12,
//    PrimeUpdate35_9DB24424FFE117E4360FB889EC00EBC4     = 13,
//    PrimeUpdate35_8B91A692CFC5CEF406CF0B7A5340F490     = 14,
//    PrimeUpdate35_A71FDE17430E717C0EA90A81E08C7D09     = 15,
//    PrimeUpdate35_F3CB1DE67976A82AF24C37EA7BF28116     = 16,
//    PrimeUpdate35_B8FB465286F92DA86FCFFABCD943FC43     = 17,
//    PrimeUpdate35_7BD907E5EE245062AD8C8911144583B3     = 18,
//    PrimeUpdate35_2F59A5E90E71DA06C07E887054F77C0A     = 19,
//    PrimeUpdate35_1EAFB7B3369A59BFE2D6047EACED36EB     = 20,
```

```
//    PrimeUpdate35_E2BFF79629FF2D72D31C3B0BF09C3D32     = 21,
//    PrimeUpdate35_END                                  = 22
//};
//
//
// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate35B
//enum class EContentKeyIndex_PrimeUpdate35B : uint8_t
//{
//    PrimeUpdate35B_97490524374FF46E3131FDF063239CE4    = 0,
//    PrimeUpdate35B_22960F06997C71815DAEE2A35FBF8BC0    = 1,
//    PrimeUpdate35B_EF057E3FB173AB964D918CAF3AF2475C    = 2,
//    PrimeUpdate35B_86379A41A2C7B95FEB8FD72CB8131592    = 3,
//    PrimeUpdate35B_E16F5C7C109AA985A927839F6F512ABF    = 4,
//    PrimeUpdate35B_8543F0D0AAD75DB7C983D461F39E56E3    = 5,
//    PrimeUpdate35B_530D464DD5C6C12AB08E881BD8D38800    = 6,
//    PrimeUpdate35B_45A9B88D52F97C76C05A176D2187E859    = 7,
//    PrimeUpdate35B_4662C606CB2FA4A93DBF972502AD7DD4    = 8,
//    PrimeUpdate35B_7697296F2773D7A7DD64E022BB837375    = 9,
//    PrimeUpdate35B_END                                 = 10
//};
//
//// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate35C
//enum class EContentKeyIndex_PrimeUpdate35C : uint8_t
//{
//    PrimeUpdate35C_05A3B69A5A7CB6AE3166DDD98B520A0A    = 0,
//    PrimeUpdate35C_END                                 = 1
//};
//
//// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate36
//enum class E_Types_Generated_EContentKeyIndex_PrimeUpdate36 : uint8_t
//{
//    PrimeUpdate36_BD02DFB4BA13F3A1C777309414664BD5     = 0,
//    PrimeUpdate36_96D8832E6CBAC01E4D931B3548C8B6CD     = 1,
//    PrimeUpdate36_922639B6EED97FC33B940DA2D78BCE64     = 2,
//    PrimeUpdate36_29FCC7E07D881E7A8BB63A1E8F4A1CD2     = 3,
//    PrimeUpdate36_731AA8F8623617B16C1BD20F8C515560     = 4,
//    PrimeUpdate36_D1AB99156B0AF5C86BD038FD5FA21211     = 5,
//    PrimeUpdate36_52008718FF3F8E5A68EFF1BDDD4FA5EE     = 6,
//    PrimeUpdate36_41C3A8A12956D299753BD860AFB7A8FB     = 7,
//    PrimeUpdate36_6AE06FAE3499CEC119EAD89D83AB2499     = 8,
//    PrimeUpdate36_5CAED8D31082564BDB859CA06D232CF6     = 9,
//    PrimeUpdate36_CC85E70DA7E6B82DBDE7E0C497B4469D     = 10,
//    PrimeUpdate36_70C46454E80360E711D3B5D043B6855D     = 11,
//    PrimeUpdate36_0231148CE41EAFEA374620374BEC872A     = 12,
//    PrimeUpdate36_1C2F3E291DFBDF4BCD89F4BBDECFE4A5     = 13,
//    PrimeUpdate36_CD94FF5FD517B8C2EBEE67C6A2F2861A     = 14,
//    PrimeUpdate36_DCCBCFB3508ACFA4A6FBA1CF9AB91B66     = 15,
//    PrimeUpdate36_6CB0303A669EA329A382223B785B54DF     = 16,
//    PrimeUpdate36_DC154AE8320F56120BE27BFAED583980     = 17,
//    PrimeUpdate36_14F6948364A30847CB208270921CBE75     = 18,
//    PrimeUpdate36_0647B0F31D68BAE45AB0E36C8C00D269     = 19,
//    PrimeUpdate36_820D6BDAA81D873F00D0EDEDC2761BA8     = 20,
//    PrimeUpdate36_3518176919EA606BF7CBC730282A7BC7     = 21,
//    PrimeUpdate36_065B320F11A9D3CC4B5E846E60893E3C     = 22,
```

```
//    PrimeUpdate36_E1369812C6C752435ABFAB3C4D67F15A     = 23,
//    PrimeUpdate36_9E47F9DB92B6D6C69E851F08DE0C89C3     = 24,
//    PrimeUpdate36_8C693F7DD88448A5DB6756E31938F863     = 25,
//    PrimeUpdate36_6E5D977724CA7C75A8B0C3BFFF28BB61     = 26,
//    PrimeUpdate36_B1D1D39BC2C8015B49D24D198243890D     = 27,
//    PrimeUpdate36_9E23CAA14A00C20AC1E137377591F377     = 28,
//    PrimeUpdate36_00043665A4DEC13AD694ABBD2609F1FC     = 29,
//    PrimeUpdate36_E9B6B7C5EDCE42EA137CB0DB7B45E914     = 30,
//    PrimeUpdate36_5B7CD65C59019880CBBD705E239D0510     = 31,
//    PrimeUpdate36_F67E4AFBB01FE7080BD18289E5EA1B77     = 32,
//    PrimeUpdate36_A4F892BC8702F12078F181987E4E9308     = 33,
//    PrimeUpdate36_END                                  = 34
//};
//
///// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate36
//enum class E_Types_Generated_EContentKeyIndex_PrimeUpdate36 : uint8_t
//{
//    PrimeUpdate36_1_467BBC80C4360BDAEF091CA61DF71FD9   = 0,
//    PrimeUpdate36_1_AC04D31B9DAD00095CE5910C5F6CE072   = 1,
//    PrimeUpdate36_1_E123A910668894ADBE430FD08A0112B5   = 2,
//    PrimeUpdate36_1_403C4336E20CC3358A7978095F21089E   = 3,
//    PrimeUpdate36_1_CA1080FB9E4F893B100A9FD89E29D72D   = 4,
//    PrimeUpdate36_1_F6FECC72E3648A6B10D9F754296B4914   = 5,
//    PrimeUpdate36_1_615756B3A90294CC4F9DC73235B67DA1   = 6,
//    PrimeUpdate36_1_0F5DB02BA90351FD4DC701F571EEC438   = 7,
//    PrimeUpdate36_1_END                                = 8
//};
//
///// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate36
//enum class E_Types_Generated_EContentKeyIndex_PrimeUpdate36 : uint8_t
//{
//    PrimeUpdate36_2_BCF72246CE09BB12CDD66EF72732FFC7   = 0,
//    PrimeUpdate36_2_06432D3A1558F6E8EA6605EAB69D66A1   = 1,
//    PrimeUpdate36_2_END                                = 2
//};
//
//
// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate37
//enum class E_Types_Generated_EContentKeyIndex_PrimeUpdate37 : uint8_t
//{
//    PrimeUpdate37_C60B25EC20D29B0FE699ACB6DFEBE550     = 0,
//    PrimeUpdate37_64DC68F90CF2281F000D1F112842BEF5     = 1,
//    PrimeUpdate37_A5E2241C686E4250022749D15398F8A5     = 2,
//    PrimeUpdate37_B3FF7B76D620718F1A46112FBBD8FDE8     = 3,
//    PrimeUpdate37_05A8B3591728E65FD3A1FC056CE5A06A     = 4,
//    PrimeUpdate37_A8C32F52DCF7B5AD13E685B30AD6870F     = 5,
//    PrimeUpdate37_241F4E6D99E40F9950EE03C50467206B     = 6,
//    PrimeUpdate37_07F1AD24B296371ACF5DA7E42385C246     = 7,
//    PrimeUpdate37_1CB1681D2D432B0DEBE23E725F8817CA     = 8,
//    PrimeUpdate37_6708BE6FD6BF625D7F89A9E941505C2B     = 9,
//    PrimeUpdate37_586FEAEB99B5F8B7AE7D59FA6AFD0526     = 10,
//    PrimeUpdate37_END                                  = 11
//};
//
```

```
//
// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate37
//enum class E_Types_Generated_EContentKeyIndex_PrimeUpdate37 : uint8_t
//{
//    PrimeUpdate37_1_A35B6EED2A82E8A6C16B36C1942DEE94   = 0,
//    PrimeUpdate37_1_E08A2E1906A83AF70BDB14C01B413788   = 1,
//    PrimeUpdate37_1_E35624A59E4478F4E4610A7EE0E3A067   = 2,
//    PrimeUpdate37_1_874FE739DDFD6DC9FE9F644BC875505A   = 3,
//    PrimeUpdate37_1_4A040F564DEAF0E711D9E76916A7BF5A   = 4,
//    PrimeUpdate37_1_92A6422919A6EE16B05FC41CB85DA4B9   = 5,
//    PrimeUpdate37_1_END                                = 6
//};
//
//
// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate37
//enum class E_Types_Generated_EContentKeyIndex_PrimeUpdate37 : uint8_t
//{
//    PrimeUpdate37_2_34241E8C41EF7AA005DD4CBB9512D6CC   = 0,
//    PrimeUpdate37_2_930CF89BB70BBACD1A197C424E596A70   = 1,
//    PrimeUpdate37_2_396DCF75DFB0A5F10580F6164CA748AF   = 2,
//    PrimeUpdate37_2_389EF40A28311D8DC25FC573202288CC   = 3,
//    PrimeUpdate37_2_8F416D7D1D05939B131ACC9EC7DB13B3   = 4,
//    PrimeUpdate37_2_19ED1A67306FBDD5E085AA3F8C6B4F90   = 5,
//    PrimeUpdate37_2_C2870E6A57B88EA7C282A05A58DEBC19   = 6,
//    PrimeUpdate37_2_D07FE80E5424327B0376EB5F13F2C6F3   = 7,
//    PrimeUpdate37_2_6E17905E8A15D170220CBEF83BC5F5E4   = 8,
//    PrimeUpdate37_2_C4E980AC9301965C4223D278329A11D2   = 9,
//    PrimeUpdate37_2_E4104F6B849F8153DACF32569DF33721   = 10,
//    PrimeUpdate37_2_AC1843530CE904B4572C07C3DFE0E118   = 11,
//    PrimeUpdate37_2_21C851D7A47C3BA8246F812FF8213048   = 12,
//    PrimeUpdate37_2_4A16D342F614F475C212E4989210B975   = 13,
//    PrimeUpdate37_2_6667BB77A92F1A9B72C66E2637832020   = 14,
//    PrimeUpdate37_2_F732C24E61CAF32BF715B312A80068F8   = 15,
//    PrimeUpdate37_2_8150D13BC1E6FA3D2720C55C906FD6C4   = 16,
//    PrimeUpdate37_2_40912F8DF7ADD15EDEC8FDB34AF6BA59   = 17,
//    PrimeUpdate37_2_C065F738DDD71599BA20FA765BFB9CC1   = 18,
//    PrimeUpdate37_2_B36D585A7AFD107C702F315A37EA2131   = 19,
//    PrimeUpdate37_2_AAA41825540F87F4D1CA59E525AE5810   = 20,
//    PrimeUpdate37_2_429E15D74DFEDA4DD1848FC05D1E15C7   = 21,
//    PrimeUpdate37_2_7A25AF1E53F2F49DEA2C1B452B84A9DE   = 22,
//    PrimeUpdate37_2_END                                = 23
//};
//
//
// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate38
//enum class E_Types_Generated_EContentKeyIndex_PrimeUpdate38 : uint8_t
//{
//    PrimeUpdate38_8D82153A816BE5FDFB750D3863064110     = 0,
//    PrimeUpdate38_3D01F0EC1174B5420EF917F8ED2FE74B     = 1,
//    PrimeUpdate38_D9F1DF2EF9878732398231D4E894EDAB     = 2,
//    PrimeUpdate38_ED3CCAA03782AE4BCC293E887B397824     = 3,
//    PrimeUpdate38_4F272FC54EA9CDF8FACC3776F7E05BC8     = 4,
//    PrimeUpdate38_23B0D661F2AE00C3E3146349AE65CA93     = 5,
//    PrimeUpdate38_C9EFEA0245734E595BA50D9D99A4126F     = 6,
```

```
//    PrimeUpdate38_286C3ABD0BFF8F0B40A6EDF7D8149451     = 7,
//    PrimeUpdate38_D90559F31BB500A1B5FA9BE8A0CF4F15     = 8,
//    PrimeUpdate38_8AFB6A24EE582323CA4270A5D96B595B     = 9,
//    PrimeUpdate38_0E4CABA177DBAA97B3690B07EC427098     = 10,
//    PrimeUpdate38_29585B8A7ACB53BA2D889BDEDC86D40E     = 11,
//    PrimeUpdate38_5362657B437EBDB2AC46BA4BC9718405     = 12,
//    PrimeUpdate38_240861E1B723C97B74325BF55B05F954     = 13,
//    PrimeUpdate38_EDDC577C8EC70EE8CBAE2EEA6AEA449A     = 14,
//    PrimeUpdate38_A5C6695C12BBA6364E306C2BD4B91B20     = 15,
//    PrimeUpdate38_END                                  = 16
//};
//
/// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate38
//enum class E_Types_Generated_EContentKeyIndex_PrimeUpdate38 : uint8_t
//{
//    PrimeUpdate38_1_34B3E2366B1725E584146D0FAD1700F9   = 0,
//    PrimeUpdate38_1_70812A493AB87B93ABECC7B119B0F6CC   = 1,
//    PrimeUpdate38_1_15D8907E96E14673950E473B42D34D77   = 2,
//    PrimeUpdate38_1_2061B4B8DD4DA4673DB2D7583F799980   = 3,
//    PrimeUpdate38_1_2E2B1ED7FE01BE3BF7CBE4BCEE050394   = 4,
//    PrimeUpdate38_1_601BA73441AFA33DFCB2A39F4745F183   = 5,
//    PrimeUpdate38_1_D866CA76AA0D2FCB12213FEEBB1ABE0E   = 6,
//    PrimeUpdate38_1_196973A75D14461F09FB8A8DDA9C06AD   = 7,
//    PrimeUpdate38_1_A5E8A2856D060F0027FA0EE9C232531A   = 8,
//    PrimeUpdate38_1_D6A221243A553D92EF9310FA078F284B   = 9,
//    PrimeUpdate38_1_A7B0D8E9B7E287731C6DEEA179034086   = 10,
//    PrimeUpdate38_1_8FFE8276FCA0677D115158A46A4F6C96   = 11,
//    PrimeUpdate38_1_FAF409B0E6620FD109E7312196D7D644   = 12,
//    PrimeUpdate38_1_8DDD441142D7C23A4483EB853F1E3D30   = 13,
//    PrimeUpdate38_1_271920F98BCD5047AEAD1793E2575845   = 14,
//    PrimeUpdate38_1_END                                = 15
//};
//
//
// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate39
//enum class E_Types_Generated_EContentKeyIndex_PrimeUpdate39 : uint8_t
//{
//    PrimeUpdate39_56421AAC43705872D01BED0B45A1B0E6     = 0,
//    PrimeUpdate39_357448C26C4CB1BB2455CE4D23B15B6F     = 1,
//    PrimeUpdate39_EFF34C512B1B73E3B3A1C5D865005AA9     = 2,
//    PrimeUpdate39_02A5906F01D11C418D125D37EA4772FB     = 3,
//    PrimeUpdate39_C9EA27ABBFE9ECB9D82E9716E8CAC6AE     = 4,
//    PrimeUpdate39_00C7978EF1D67EE30874405F1EE7D085     = 5,
//    PrimeUpdate39_50936FD9197A5FC47E5CD7DC265137C8     = 6,
//    PrimeUpdate39_E018D55157E0C905FF33BC28F2B9D98C     = 7,
//    PrimeUpdate39_9CC7DDEE9F2A4BBD06189C80559B7BBD     = 8,
//    PrimeUpdate39_6C37D54A8F53F128B70B58FF97241716     = 9,
//    PrimeUpdate39_ECA424488B0B7C94307B7745DF958979     = 10,
//    PrimeUpdate39_8D577B8AC9F1B9BC3614B6DB90B37C5B     = 11,
//    PrimeUpdate39_AD198FBCFFFB2E31230F545EDB9065D0     = 12,
//    PrimeUpdate39_7F8192404C707DE4128174751FAF1D00     = 13,
//    PrimeUpdate39_B749D533EDC231DCDA92FE5375FC6589     = 14,
//    PrimeUpdate39_AB6E50152C4025F6142036BD74016C7F     = 15,
//    PrimeUpdate39_705F95D329B744F6D26A4CF94D85A211     = 16,
```

```
//    PrimeUpdate39_523F85BCAB19FC6B10F6066203D67DE6     = 17,
//    PrimeUpdate39_324C031A7F1C4366354A79E5A4AF0086     = 18,
//    PrimeUpdate39_END                                  = 19
//};
//
//
// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate39
//enum class E_Types_Generated_EContentKeyIndex_PrimeUpdate39 : uint8_t
//{
//    PrimeUpdate39_1_5E8D9027C3059552E1664170E791A1B3   = 0,
//    PrimeUpdate39_1_3BEAE926AE33C0BF8F95782CB4B36A4A   = 1,
//    PrimeUpdate39_1_9A3C1EC3957DF2AA96A2AF2DC22BA3FC   = 2,
//    PrimeUpdate39_1_FD16CE749738F92F5A140AB7C7AC959E   = 3,
//    PrimeUpdate39_1_D86B0AA89DEB300AA7EEF30FD07ABFD7   = 4,
//    PrimeUpdate39_1_92158C7B6826AC37F146AF5D5B4AC9DE   = 5,
//    PrimeUpdate39_1_792A401E32DC3DB0AFE00611C35572BD   = 6,
//    PrimeUpdate39_1_1A067C16B1DF704EC9AE6B921C8DE4AE   = 7,
//    PrimeUpdate39_1_C905548B21885467611F998CAC53EEFB   = 8,
//    PrimeUpdate39_1_21DE447D1209866593DDFF980DA7C147   = 9,
//    PrimeUpdate39_1_6515064E9C27EF5B0688BADD7318CB03   = 10,
//    PrimeUpdate39_1_1F8D1D27969AFDB0DF4560A2E4E16030   = 11,
//    PrimeUpdate39_1_5F925119F81B86A64DCF8461CE729846   = 12,
//    PrimeUpdate39_1_BDF4A8BBCE902498FA3525F29D728C44   = 13,
//    PrimeUpdate39_1_7A65FF032B288BAB9C23958AE0054266   = 14,
//    PrimeUpdate39_1_83BA8046B2F752EB6CFF4203BD7DF88F   = 15,
//    PrimeUpdate39_1_AE07A44FEB585B27188B1C4D8D2D1C3D   = 16,
//    PrimeUpdate39_1_CDFB3137294E4278E07C70B780DF4B2E   = 17,
//    PrimeUpdate39_1_459B4AE9216EF3FE102EF227BA7F8E62   = 18,
//    PrimeUpdate39_1_F754BFA3BC83BE9DD8F1FDA1D7881BF3   = 19,
//    PrimeUpdate39_1_END                                = 20
//};
//
//
// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate40
//enum class E_Types_Generated_EContentKeyIndex_PrimeUpdate40 : uint8_t
//{
//    PrimeUpdate40_8C2237625A781A11D2F3F89FEB5049E6     = 0,
//    PrimeUpdate40_346FFB9D30A3C98CF6E26049C75F8A6F     = 1,
//    PrimeUpdate40_5F8C0631E0B7B9DFC2F3008C3745E87B     = 2,
//    PrimeUpdate40_7FE4E4CE92A9C8467009C6BCF579BA99     = 3,
//    PrimeUpdate40_278DC229F3110DD6AB3267F83EB1B495     = 4,
//    PrimeUpdate40_7FC56CB1544D4E7D994C896BA3218FD0     = 5,
//    PrimeUpdate40_3D88BFDE61D0306701BCC41860135271     = 6,
//    PrimeUpdate40_01E2A30D1ADA99F14F4E8EEFBD6B42F9     = 7,
//    PrimeUpdate40_229839877715A14FFD231E2A91671E05     = 8,
//    PrimeUpdate40_44DE38F6740A856E177316F57C7B6C20     = 9,
//    PrimeUpdate40_0602631BBDAAE3DE752DB076B9980677     = 10,
//    PrimeUpdate40_02403FEAADC79FFFC4B35E5658A00545     = 11,
//    PrimeUpdate40_845A3762DBC7CA41402447CF2556D4C7     = 12,
//    PrimeUpdate40_53D7FB6D2E642F86A3A075F6FBB1786F     = 13,
//    PrimeUpdate40_6B6EA27864353D11035CB81228DC0F66     = 14,
//    PrimeUpdate40_388A578A38C54067CE555AC93F4AEBB7     = 15,
//    PrimeUpdate40_FDCAFEA62CCFA9C90C79D21B2071D2A3     = 16,
//    PrimeUpdate40_A82F8D3689507CB67E8D7D52853B932F     = 17,
```

```
//    PrimeUpdate40_B98DE12096263666327D7524E3841577     = 18,
//    PrimeUpdate40_33A533A07A82D47AF416BE38E17C89D8     = 19,
//    PrimeUpdate40_A747A04581ADEBCA8676A7D75BB9686F     = 20,
//    PrimeUpdate40_D5860C3628E2C0435CD53BECAA795728     = 21,
//    PrimeUpdate40_C464F90B96CC841599F40371C84AB72D     = 22,
//    PrimeUpdate40_183FC665FE74AC5907F6D056803596A2     = 23,
//    PrimeUpdate40_8696BE011AD3EA49278489597BCD9EBB     = 24,
//    PrimeUpdate40_4BB762B822928F39C7828FEC4DB7A93A     = 25,
//    PrimeUpdate40_C3B704181B79C72144ED4AB74CAA2851     = 26,
//    PrimeUpdate40_7BF9ED88B50F35D452FE4248C1A253FB     = 27,
//    PrimeUpdate40_5BC35E8AE7B39F853713CE1A071F4ACD     = 28,
//    PrimeUpdate40_F9D65D20485C4F6305EF8712F06A6EA8     = 29,
//    PrimeUpdate40_9B1E2298F76498A3AA1994061533747E     = 30,
//    PrimeUpdate40_8ED9D5B5C91C2DE1E5BADE5F64C7C02A     = 31,
//    PrimeUpdate40_7B0FDC201F51B0B00AD1FD4969066042     = 32,
//    PrimeUpdate40_END                                  = 33
//};
//
//
// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate40
//enum class E_Types_Generated_EContentKeyIndex_PrimeUpdate40 : uint8_t
//{
//    PrimeUpdate40_1_250828A4E25377F0718BF942E3FB9017   = 0,
//    PrimeUpdate40_1_C4077BE4699B9B2D99DDC6DE7722DA09   = 1,
//    PrimeUpdate40_1_9649CE040FC2F9F6FC25B25A0CBF10D8   = 2,
//    PrimeUpdate40_1_BEEB4A30886DA7D8C011C0DF4179C28A   = 3,
//    PrimeUpdate40_1_1B295026CE2F4E8732BFDBC57E9545C7   = 4,
//    PrimeUpdate40_1_C37309958D55A38BA76976651F85DB64   = 5,
//    PrimeUpdate40_1_EAA0F03C8B7809EF8E4D4ABC622C8323   = 6,
//    PrimeUpdate40_1_840309E0442EB9B93564AAF65B94DAB2   = 7,
//    PrimeUpdate40_1_825EFFBF370C154712F6CFA2C44BE6BF   = 8,
//    PrimeUpdate40_1_B8841459B9499F5F0632B0C753212E13   = 9,
//    PrimeUpdate40_1_5642DF159160EEE1408B8E3583BF2CF4   = 10,
//    PrimeUpdate40_1_4BE1E668C754F1F1C404F27615D0F080   = 11,
//    PrimeUpdate40_1_8B3D592961D1F9395CC3ACE70436D7FE   = 12,
//    PrimeUpdate40_1_67C82A8E453258CF014D5BE2A65F85AB   = 13,
//    PrimeUpdate40_1_560403CDAB63565DE5113D3B5613A3BA   = 14,
//    PrimeUpdate40_1_0C4A48343FB45CD451885E6631FEDF5C   = 15,
//    PrimeUpdate40_1_END                                = 16
//};
//
////// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate41
//enum class EContentKeyIndex_PrimeUpdate41 : uint8_t
//{
//    PrimeUpdate41_FCF4AC8D3926122F728684A3A7CE9BD0     = 0,
//    PrimeUpdate41_FE412C244D9C4AE088AA24BAEE47064A     = 1,
//    PrimeUpdate41_1163DFB4A5200F5D720D05032D45F072     = 2,
//    PrimeUpdate41_F2BE58615CBFD9E7625BDF764CB83A3F     = 3,
//    PrimeUpdate41_816C46ED6F43BA8EE2B9D6D4114ECB61     = 4,
//    PrimeUpdate41_736C408761D627AFD02780C644C41A4B     = 5,
//    PrimeUpdate41_E5C03F083C3ADF00C3A717A945514920     = 6,
//    PrimeUpdate41_99072B23797227CD12055FAD972518C9     = 7,
//    PrimeUpdate41_AF629C34B96FEEF9A9CBFB864E6B2CD8     = 8,
//    PrimeUpdate41_8CED5FD39030435913600AAD6FE9FF15     = 9,
```

```
//    PrimeUpdate41_0E60B37DF8579D73C651AE2E8D4E11A6     = 10,
//    PrimeUpdate41_52DEE812DA1533B1C4F2C82B17B45CBE     = 11,
//    PrimeUpdate41_46F30185A7CE929CB6EEC79A343506EE     = 12,
//    PrimeUpdate41_1A29E290ACEEB4E686F5D2FB4D8F8AA7     = 13,
//    PrimeUpdate41_E3DF6729271F9223B39EFC623A2A1F11     = 14,
//    PrimeUpdate41_C16726300EFA42B243676ECEEFA3C96F     = 15,
//    PrimeUpdate41_END                                  = 16
//};
//
//// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate42
//enum class EContentKeyIndex_PrimeUpdate42 : uint8_t
//{
//    PrimeUpdate42_E52531C6B1B6F115D558C6139F218E25     = 0,
//    PrimeUpdate42_A5964FEE8B1B9A5D51E457A687F438E1     = 1,
//    PrimeUpdate42_8927A460BC8CB874B64839B7542CD145     = 2,
//    PrimeUpdate42_2E2B46230BDFD4F2E537F0031212E27B     = 3,
//    PrimeUpdate42_B50B693A275494257E9B74EEDCA19CFA     = 4,
//    PrimeUpdate42_50EC783373F1547A15538556F509EB67     = 5,
//    PrimeUpdate42_D9DF25EBE864C5225992E9DAA5BEC7D4     = 6,
//    PrimeUpdate42_B69AB6BDE915FCB8E4F97B127D34D846     = 7,
//    PrimeUpdate42_DCA80A3990A81EE62FAAB810A6120F7C     = 8,
//    PrimeUpdate42_0C0343D9BC8C40BD3A79E4BC2079D094     = 9,
//    PrimeUpdate42_ACF6AA89B69C8CC7167F359805B47808     = 10,
//    PrimeUpdate42_B6F39F5D7E72BF64879EA05F6ED2E6D4     = 11,
//    PrimeUpdate42_3F6C2706DDB94EC590D4DAB3545067F5     = 12,
//    PrimeUpdate42_2193CC48D5CF93AF688E850F52B4451D     = 13,
//    PrimeUpdate42_DDF9482199E02E6566C9CAEA28C4EC62     = 14,
//    PrimeUpdate42_FE5DA93948F7C2140E43DA64811F467B     = 15,
//    PrimeUpdate42_2C9C5B90D28216CF9A81AFD43E63AF93     = 16,
//    PrimeUpdate42_51A0A9E9531104F14E0F63EA6F012D78     = 17,
//    PrimeUpdate42_END                                  = 18
//};
//
//
// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate43
//enum class EContentKeyIndex_PrimeUpdate43 : uint8_t
//{
//    PrimeUpdate43_CC671B6037A1D59F3C1821B7233DFF04     = 0,
//    PrimeUpdate43_2C3CFF07FE767FF60F6D6DC087173AB4     = 1,
//    PrimeUpdate43_F49C0E2E4A8F4D683856334D067C6F5D     = 2,
//    PrimeUpdate43_76670877373BB64A7CD24AF0138BB295     = 3,
//    PrimeUpdate43_810FDE30237521E13B560D507CA768B3     = 4,
//    PrimeUpdate43_E66D8C6CCA95C91F8918DC7562D1EACB     = 5,
//    PrimeUpdate43_52C29922C0E4EB2BFD4E13D940FA5218     = 6,
//    PrimeUpdate43_5616A28B9859ADDF538D11E7DF1979AA     = 7,
//    PrimeUpdate43_3EE780061C5061F71D3E75AD181C2106     = 8,
//    PrimeUpdate43_11BE14B80EBD87E5834349696E23276C     = 9,
//    PrimeUpdate43_25F4690FAC8C10F30940CA45581F2E50     = 10,
//    PrimeUpdate43_89B5BF6B6E2356454D75AFF944E84AEC     = 11,
//    PrimeUpdate43_F786E11A2578748A2279BA1C872068DB     = 12,
//    PrimeUpdate43_606B443C783353BBA80B23F4D2358B99     = 13,
//    PrimeUpdate43_649D0E0445D3EAD3ED3458293369F915     = 14,
//    PrimeUpdate43_6C28AA72305A66002E60EB698956D6E3     = 15,
//    PrimeUpdate43_63F063DB73E258A05B51B5F88DB913A2     = 16,
```

```
//    PrimeUpdate43_32CA5400E5D0A5F7FF0B79211D6F1D15     = 17,
//    PrimeUpdate43_040B1384D3406947830105729DE06007     = 18,
//    PrimeUpdate43_6F9A44FA71B4D98EB2C05867D5C09545     = 19,
//    PrimeUpdate43_2332FBF310DAC3DBAC0AB1A435B92843     = 20,
//    PrimeUpdate43_1A8C58CEE6B233247AF7574689031FCA     = 21,
//    PrimeUpdate43_042D95D22673F2872277B93F921D0E03     = 22,
//    PrimeUpdate43_0E043AA25A9763DEF3ABCDB4CA04B40B     = 23,
//    PrimeUpdate43_7640592175EED2834F2A92A752DC9395     = 24,
//    PrimeUpdate43_END                                  = 25
//};
//
//
// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate44
//enum class EContentKeyIndex_PrimeUpdate44 : uint8_t
//{
//    PrimeUpdate44_1446BABD70B69A712C133030AE513B21     = 0,
//    PrimeUpdate44_71AF65550CA5ADF69C006E7B97F650B3     = 1,
//    PrimeUpdate44_5DEEFF412F4BA7EBA3CC0FC5C573B6A1     = 2,
//    PrimeUpdate44_077DD29252345C70A22D50C17FB62366     = 3,
//    PrimeUpdate44_598106981F888616B81601A887615564     = 4,
//    PrimeUpdate44_CB2633DDA5919383239889125778575C     = 5,
//    PrimeUpdate44_E51BDCC741F8B25643EFFD71C7004E8E     = 6,
//    PrimeUpdate44_734F8579680535EB0DEACC8DF366B2D7     = 7,
//    PrimeUpdate44_1840F1E0254207DA2867FC7C2AE6D61A     = 8,
//    PrimeUpdate44_4FD312E0A9015EFE89F61154FC1DE885     = 9,
//    PrimeUpdate44_AD752B0369DE162C51A8680F088E123D     = 10,
//    PrimeUpdate44_C90FAAD83F369944C02A265506DA0053     = 11,
//    PrimeUpdate44_7809BFF8943D2CB0DD60C19430DC30A2     = 12,
//    PrimeUpdate44_B98F7016457ADFEFEFEBEFD2E08E2B20     = 13,
//    PrimeUpdate44_98017FA7C261C53FF46DB08F25160AAF     = 14,
//    PrimeUpdate44_76D1B56D70961C57344657F7E8E3DE3F     = 15,
//    PrimeUpdate44_END                                  = 16
//};
//
//
// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate45
//enum class EContentKeyIndex_PrimeUpdate45 : uint8_t
//{
//    PrimeUpdate45_B5EAA36499C5256F6DF4EBD1FE200CCF     = 0,
//    PrimeUpdate45_0933157559707C5150C6AF7CEF4AAC17     = 1,
//    PrimeUpdate45_AEC0609A22360A523E8F12D1D93392B6     = 2,
//    PrimeUpdate45_6021CF109BB13FA0178773921CC56325     = 3,
//    PrimeUpdate45_6D50775660E0EAE0D40DF4CAD9278285     = 4,
//    PrimeUpdate45_2B2B580AC3587A4EA34B21658122F190     = 5,
//    PrimeUpdate45_6D685056C5B9236E3F0BE8255810FD1F     = 6,
//    PrimeUpdate45_54939A69E4EE576DC9C585A81EF6C663     = 7,
//    PrimeUpdate45_0B73B2E2A38A1FCE42F1B92AFD23537F     = 8,
//    PrimeUpdate45_1C5D3D58F7927BE303B809D89F7143D3     = 9,
//    PrimeUpdate45_BC9C9F38082C6791C8E549D22869DE08     = 10,
//    PrimeUpdate45_C381FBF2F67E1AAA6A2E295F5625A5F6     = 11,
//    PrimeUpdate45_CAF8D1B42134C226F1DA07A708674278     = 12,
//    PrimeUpdate45_1B8E4344D969AED0426DC2DFA4B72ADC     = 13,
//    PrimeUpdate45_0053FEEA7307DA7AFA2A7B7E35BC49B5     = 14,
//    PrimeUpdate45_10D93F18596502C09021729CBD5B889D     = 15,
```

```
//    PrimeUpdate45_6B088106B25FD4E5F6BC43EAF5100F95     = 16,
//    PrimeUpdate45_6D3C8A03CFD878CECAB4B5AC5C03CAB6     = 17,
//    PrimeUpdate45_3FAEE4E6FEEDA84F8769F871A4A642E0     = 18,
//    PrimeUpdate45_0836DB4FE635A5907BA5D850E48BB6A4     = 19,
//    PrimeUpdate45_46D96ACE6F06DCC43948F15F5AECC228     = 20,
//    PrimeUpdate45_0F236B3CBC14F466B2C4E9C8BB4DC6A6     = 21,
//    PrimeUpdate45_END                                  = 22
//};
//
//
// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate46
//enum class EContentKeyIndex_PrimeUpdate46 : uint8_t
//{
//    PrimeUpdate46_1158992028E7AFBF9A18D7162C53E469     = 0,
//    PrimeUpdate46_0631F1BFE7C5CBE5D74702C3C4CA672C     = 1,
//    PrimeUpdate46_D70E6236296B8E83C597337C8787074A     = 2,
//    PrimeUpdate46_8545088C2EF44257E47BECD6686E9410     = 3,
//    PrimeUpdate46_A4EC79CD8BD95B88EFFDE529B4473BDD     = 4,
//    PrimeUpdate46_259BD958CAD1EBD253F65EF94D0C2BCE     = 5,
//    PrimeUpdate46_8C083608FA180E6F8A002A727506299B     = 6,
//    PrimeUpdate46_7176F1EE1BA95EB10EBFA17712C10403     = 7,
//    PrimeUpdate46_E055615D21B2ACEDDFFACDECF75C5035     = 8,
//    PrimeUpdate46_737F86BD663C08C4FD04BD68AAE5E9E7     = 9,
//    PrimeUpdate46_070A8FB59E49C3B7DD68CB402D2D99C2     = 10,
//    PrimeUpdate46_92E18474A103766860D7153FE87F132B     = 11,
//    PrimeUpdate46_1A00ECA8948D81F67D12D3256C0FF183     = 12,
//    PrimeUpdate46_9B2F1C064EB5CFFFE5B514A472D71AFF     = 13,
//    PrimeUpdate46_9F90C18737616F81EBD41574E133B5C3     = 14,
//    PrimeUpdate46_5BEB2823CFEBE00C153E21E1EC858A11     = 15,
//    PrimeUpdate46_0703F1A7CEF5CEFA191F954B89E70878     = 16,
//    PrimeUpdate46_124BB2A7BE71920A1E89D96EF3221D21     = 17,
//    PrimeUpdate46_2A40072A42F9D94354A3F088AA63252F     = 18,
//    PrimeUpdate46_END                                  = 19
//};
//
////// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate47
//enum class EContentKeyIndex_PrimeUpdate47 : uint8_t
//{
//    PrimeUpdate47_309E904795840EC71FAD9655E737C15D     = 0,
//    PrimeUpdate47_74D1CE2ECE537AD9A71E6A2F4E397036     = 1,
//    PrimeUpdate47_32E55C0B969118CDE148192DBCBF1D8A     = 2,
//    PrimeUpdate47_4B2D0A5AC7D8646E341087D0881200FC     = 3,
//    PrimeUpdate47_7373B3C7AE89E56050B3D8D5DE47F638     = 4,
//    PrimeUpdate47_0F195D246CAD9C9D611FFF1FB81E17FE     = 5,
//    PrimeUpdate47_6E71DF78D722B23043466181BB7BADDA     = 6,
//    PrimeUpdate47_CABC11491CD7DF325FD88C2D3D9AB1B3     = 7,
//    PrimeUpdate47_7D2414904BA213CF408EAD5271829219     = 8,
//    PrimeUpdate47_5259FCAB0F1C6D56775B9BCC5574DAF7     = 9,
//    PrimeUpdate47_A0AD83BC5268E59CEBCE7C70F69C8122     = 10,
//    PrimeUpdate47_08AA50461A453460D0F3308048B36A60     = 11,
//    PrimeUpdate47_49122314B316B2A05C4E3F5E46949DE8     = 12,
//    PrimeUpdate47_15234CD758D8A3B2A50BCD492F527691     = 13,
//    PrimeUpdate47_88DB5A9D96BB8E542F09C5FD047B8785     = 14,
//    PrimeUpdate47_1CDAEE409E6AE356FFFCB907ED937002     = 15,
```

```
//    PrimeUpdate47_5EFEA7DE273FE3F02537F8EE2B380AB0     = 16,
//    PrimeUpdate47_52E8F11E3E38AB5C075F5A83F1F0E824     = 17,
//    PrimeUpdate47_CEE53EAF2C7012DC1F5276A352DB3999     = 18,
//    PrimeUpdate47_BAB337A3CE44F9E0F125003B065E7FA6     = 19,
//    PrimeUpdate47_06B2A87C87DEB3324E3D0BC4FAF6795C     = 20,
//    PrimeUpdate47_END                                  = 21
//};
//
//
// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate48
//enum class EContentKeyIndex_PrimeUpdate48 : uint8_t
//{
//    PrimeUpdate48_3EF8CE740B23D9322957CD48847B36DD     = 0,
//    PrimeUpdate48_29128550512066E7172477F6A80C69B2     = 1,
//    PrimeUpdate48_C55289FC8C940342E9D20A84445F8DFB     = 2,
//    PrimeUpdate48_30C3084A1A4181BEAF61949398739D55     = 3,
//    PrimeUpdate48_A838F1BE39577AC83B7D07C75E671E95     = 4,
//    PrimeUpdate48_30287A8B3092A967E33949D90EE406B5     = 5,
//    PrimeUpdate48_D0E4DE25BB880DB72CE076DECFC8349D     = 6,
//    PrimeUpdate48_F8682AEA04D02168AE986FE0E3CA4C1F     = 7,
//    PrimeUpdate48_3B5AE0A28D5DA9403AFC72705A2B06D9     = 8,
//    PrimeUpdate48_F7B1CCDC7AEB4243E695195EDA59A427     = 9,
//    PrimeUpdate48_369188F7D83F2361C61D0A6E2A7C158F     = 10,
//    PrimeUpdate48_87D41C6EAC66133C48C5647211AC9D8A     = 11,
//    PrimeUpdate48_2AD4BFCA44C46FACF460BFD2804FD452     = 12,
//    PrimeUpdate48_90E17EDF3C2CAAC083AF72F72C683CE7     = 13,
//    PrimeUpdate48_END                                  = 14
//};
//
//
// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate49
//enum class EContentKeyIndex_PrimeUpdate49 : uint8_t
//{
//    PrimeUpdate49_E3CFC8EF5DA3E92CAAB56A9DAA6A67FC     = 0,
//    PrimeUpdate49_55EF0C693E6D73F41302C6314262AD09     = 1,
//    PrimeUpdate49_0E2831103AFF5ECAB66919B26D1C0E00     = 2,
//    PrimeUpdate49_89E6E5BC20EC9EFFBFE366E485FD0CB0     = 3,
//    PrimeUpdate49_6D7EDCB2226CC7415CE20AED0CBF55E8     = 4,
//    PrimeUpdate49_79F27DF8758E75A94E8047D17ACBD4AE     = 5,
//    PrimeUpdate49_A0E60758FF958270A430688A23A58F98     = 6,
//    PrimeUpdate49_BCCDC81A6422981D95A41987D19AC6B7     = 7,
//    PrimeUpdate49_0D94716C9A62DD2084F359935AE350D3     = 8,
//    PrimeUpdate49_C8D577410C1759E650CB2A226B272C2A     = 9,
//    PrimeUpdate49_DFC1A8BD6A9946A72C0B74815F600313     = 10,
//    PrimeUpdate49_D893E26A3ED335C708322793CB96E7C2     = 11,
//    PrimeUpdate49_F0C7BF0925F7987548EFD91439B7E923     = 12,
//    PrimeUpdate49_C8350BFC1CFEC95F55CA7F249B99C188     = 13,
//    PrimeUpdate49_74017C86D6044A055E0FB9470ADE30A5     = 14,
//    PrimeUpdate49_F9332D1CB2B08C2EB7AEB9C2F07024A0     = 15,
//    PrimeUpdate49_A63922D1764CB8C56788C193F98CF300     = 16,
//    PrimeUpdate49_1968ABBBDE1EAC29E557C42B5D131F1B     = 17,
//    PrimeUpdate49_B32A36A8914E34FBA23A7DE847EF33B0     = 18,
//    PrimeUpdate49_DADEB2A00E44A6F700AD29DB615F8A7F     = 19,
//    PrimeUpdate49_851D0409174FD6B5DF61A0BE757D5260     = 20,
```

```
//    PrimeUpdate49_11A5BACDF704ECB3E75F3D5D7922354D     = 21,
//    PrimeUpdate49_D2177317A57AED45BFE1C984F4038426     = 22,
//    PrimeUpdate49_C18C9419579438D9660096F83F61226B     = 23,
//    PrimeUpdate49_773BB92CAB0A792C57533A123136A3EA     = 24,
//    PrimeUpdate49_9BA89F72D9113A471F6566D1CBA79C03     = 25,
//    PrimeUpdate49_B5A7598C5E70D75D2BF7CF17F3D79F98     = 26,
//    PrimeUpdate49_503F5A33B4F8817BA01CBCFE0B90DAFA     = 27,
//    PrimeUpdate49_ACAC4EE3812E074FDC2E985AAA3F8E5F     = 28,
//    PrimeUpdate49_18245F939BC49D9F3DCAD14F69BD9EDA     = 29,
//    PrimeUpdate49_3759A7DB5333E76179555B7D0418E5A9     = 30,
//    PrimeUpdate49_6536D1AD86639C8E47BB05AF3F98842D     = 31,
//    PrimeUpdate49_END                                  = 32
//};
//
//
// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate50
//enum class EContentKeyIndex_PrimeUpdate50 : uint8_t
//{
//    PrimeUpdate50_F3D1F836D2AB4A4DAE38A3820EF42536     = 0,
//    PrimeUpdate50_071FCF1F6B53D3C4B73DE5700CFC4E32     = 1,
//    PrimeUpdate50_3AF0C7A0476E35569C49C19613DEC950     = 2,
//    PrimeUpdate50_078044D4CCF576219097B56F43F4592A     = 3,
//    PrimeUpdate50_8D546A00D7D4F875CD41682AA4742E4E     = 4,
//    PrimeUpdate50_2848A6873428E1973EE562B1DA375A0D     = 5,
//    PrimeUpdate50_CE2CDB2BBD868F853655FA81857607FA     = 6,
//    PrimeUpdate50_AB84FCB94926338CA5D33512CA095C99     = 7,
//    PrimeUpdate50_C623ECE9F4332FE67320376631056AA9     = 8,
//    PrimeUpdate50_2764230C6577DB9A7F73806EA7B4EB65     = 9,
//    PrimeUpdate50_6A08669DE9F305204ABB96F039A8F571     = 10,
//    PrimeUpdate50_C7F546C68229DEACFC577355ABE3A093     = 11,
//    PrimeUpdate50_CB2477FB783B5D0352AC661F03349B60     = 12,
//    PrimeUpdate50_BF9DBFFF3C2F942709622AAB37201C04     = 13,
//    PrimeUpdate50_2FE277D22BFEFF02DF6B12191E5BBBE3     = 14,
//    PrimeUpdate50_66D880FE5CEC47D0B56C087CEE88E1E5     = 15,
//    PrimeUpdate50_9FE6F44F321FC6A700490CBD6BB04613     = 16,
//    PrimeUpdate50_1DA79252C09A01B045FA2FE915923C0D     = 17,
//    PrimeUpdate50_544585EF0025E5083CA4326DDEF3655D     = 18,
//    PrimeUpdate50_F95190FC717AF94B6990942233E801FE     = 19,
//    PrimeUpdate50_11DE8FC8FC1BD4FE96831040496E0707     = 20,
//    PrimeUpdate50_4DC4EF0E5F6B3FA99241985EA91CF149     = 21,
//    PrimeUpdate50_D873B503F038FB4BEBA4543DBEF12220     = 22,
//    PrimeUpdate50_CE13BD9CF3291173F19434B573971494     = 23,
//    PrimeUpdate50_611027D90AC801CD0BEEA765469DC24D     = 24,
//    PrimeUpdate50_9D93359E3796F8C587EBE9F68B700C86     = 25,
//    PrimeUpdate50_441FEE23097BAF580714605D74A0E82A     = 26,
//    PrimeUpdate50_2084F788C5628F1E715A9D5362112800     = 27,
//    PrimeUpdate50_192CFB62EDB9CAE26F6F0F8A1DDBDADF     = 28,
//    PrimeUpdate50_6687B83178E7CBCE0CADB50C09033880     = 29,
//    PrimeUpdate50_51B4120CF5A04FBD110A0675E4813D3D     = 30,
//    PrimeUpdate50_47F4C4F07913803ED0A57D8AC4326382     = 31,
//    PrimeUpdate50_9AE526DBA07B2752A38D8D02E57CC1ED     = 32,
//    PrimeUpdate50_2CF20DA5A945E97120970A7D69EF30DF     = 33,
//    PrimeUpdate50_8288CCBFF3B9FE2BEFC038AFBB079CF5     = 34,
//    PrimeUpdate50_3A0A616CAF8A4BB78ADA62D9B8951EA3     = 35,
```

```
//    PrimeUpdate50_END                         = 36
//};

// Enum Core._Types_Generated.EContentKeyIndex_PrimeUpdate51
enum class EContentKeyIndex_PrimeUpdate51 : uint8_t
{
PrimeUpdate51_7060CB1DB66B07E8CBE3E6AAA3DC454B     = 0,
PrimeUpdate51_5E459EEB37CCF7C83748272B436D1D71     = 1,
PrimeUpdate51_0D576F3EBAAD4BB6CA7AE07746B95412     = 2,
PrimeUpdate51_D12696009BD16438B467D80BF5A076D0     = 3,
PrimeUpdate51_CDFC4EF93A9140954C851273092F9933     = 4,
PrimeUpdate51_627D56F1DEF28CED719B6E6C813680FA     = 5,
PrimeUpdate51_0A6F1A11F899707585A0B4CA967A420C     = 6,
PrimeUpdate51_93D25F24A29964A633C2427B644EF2B2     = 7,
PrimeUpdate51_74B8471339971E06535E05EF71C69115     = 8,
PrimeUpdate51_77046A1684701EE5274E90C50B6AD1AC     = 9,
PrimeUpdate51_B8C139C644F513969816473E7B889678     = 10,
PrimeUpdate51_C6174FA1231CFA7CEAB7ECECD437568F     = 11,
PrimeUpdate51_A54671833BCDD6E98C438E0832719B1E     = 12,
PrimeUpdate51_7C4D5294E866C6B7DBEC9D5854871A54     = 13,
PrimeUpdate51_7FA12DF8D1E4A759F429765B34FD6670     = 14,
PrimeUpdate51_041169E1E5A539744C254245E5F486C0     = 15,
PrimeUpdate51_CDB06895A851F961C0E1660D55042C49     = 16,
PrimeUpdate51_C88BCC35EA163234A0531E4107032FDE     = 17,
PrimeUpdate51_290D923ED0F506910EB8B699443649A9     = 18,
PrimeUpdate51_CF44EA9C716C51BACFA562E98EC359AA     = 19,
PrimeUpdate51_099E362A68FCFF6732CF71306075C707     = 20,
PrimeUpdate51_0F7DBDD2C5293EF40E324B19C47639B7     = 21,
PrimeUpdate51_5E9E9BA1D3F5E63A6A1CF62C2D3B21A3     = 22,
PrimeUpdate51_61A1D7AB38AA2F85EBD574B1C60FFB05     = 23,
PrimeUpdate51_D591A227D318ABAAA1789E38E5B98A0A     = 24,
PrimeUpdate51_98B0889F8CD2140B68268FA5ED077C7B     = 25,
PrimeUpdate51_8AFE3502E09D46361EC069D8610BBA0D     = 26,
PrimeUpdate51_0F5B3D454E9F8C0B31301ADF020105F5     = 27,
PrimeUpdate51_1C879C73D32B7BC9A7C073D52E300B7F     = 28,
PrimeUpdate51_FF7265649A1BD8FF3314AB74ECA727D8     = 29,
PrimeUpdate51_7F1C63549A526FF871C16F27DB278EBB     = 30,
PrimeUpdate51_70D9EAC680907ED2E1C75B9014AEA368     = 31,
PrimeUpdate51_5B1CB75C8D964A8BD24942B3E539DB8B     = 32,
PrimeUpdate51_B8C5319A8FD200FA9714C4129B3685CD     = 33,
PrimeUpdate51_BC41C0B66B8F535BCC7542E1207D7FA1     = 34,
PrimeUpdate51_658CEAE71B10B3AEB38A184D2BD6F9F7     = 35,
PrimeUpdate51_341F7EAD04BA71B44B9DD778AF4566F6     = 36,
PrimeUpdate51_6C8A8EAC8A71BCF48ED54C2DFD5E014E     = 37,
PrimeUpdate51_D2269297F5331CF16A37AADE5B856025     = 38,
PrimeUpdate51_2A0B1093AAE4A007F0C347F21B529A30     = 39,
PrimeUpdate51_3B1F7AA04D300A74B6EEB30993AED9F4     = 40,
PrimeUpdate51_8D084D943B1969B7F5564FCD5EDB6678     = 41,
PrimeUpdate51_C2B7DE943C4C286C25A97178A8B63311     = 42,
PrimeUpdate51_56EDDA5440735216B8235124F7AFC5A3     = 43,
PrimeUpdate51_48A2E63062D69D4522F2A41F844B6E82     = 44,
PrimeUpdate51_8D30830DACA89B85BEB9593A85D799F5     = 45,
PrimeUpdate51_BF35D02A6E25F0A14B121BE579A7C32A     = 46,
PrimeUpdate51_50CB10F941DA7DCF8B7FB90300351704     = 47,
```

```
PrimeUpdate51_2F1B6BE2AB8FB8B5C4DA65A4BA8DFE01     = 48,
PrimeUpdate51_DC9694E8FE5D979E827AD61388B7A1D5     = 49,
PrimeUpdate51_C4C31ABC826B949D3E012068791ED1B4     = 50,
PrimeUpdate51_END                                  = 51
};

// Enum Core._Types_Generated.EContentKeyIndex_ContinuousIntegration
enum class EContentKeyIndex_ContinuousIntegration : uint8_t
{
ContinuousIntegration_A21E529632046B5DAA3373A6051D7164 = 0,
ContinuousIntegration_8C0B2C5877659E4548B294EA142D4C7A = 1,
ContinuousIntegration_BC2E369B178A16B81F7B990426A8D59F = 2,
ContinuousIntegration_0834F4083483791F04893BE705044600 = 3,
ContinuousIntegration_END                          = 4
};

// Enum Core.DistributionVector.EDistributionVectorLockFlags
enum class EDistributionVectorLockFlags : uint8_t
{
EDVLF_None                                         = 0,
EDVLF_XY                                           = 1,
EDVLF_XZ                                           = 2,
EDVLF_YZ                                           = 3,
EDVLF_XYZ                                          = 4,
EDVLF_END                                          = 5
};

// Enum Core.DistributionVector.EDistributionVectorMirrorFlags
enum class EDistributionVectorMirrorFlags : uint8_t
{
EDVMF_Same                                         = 0,
EDVMF_Different                                    = 1,
EDVMF_Mirror                                       = 2,
EDVMF_END                                          = 3
};


/*
#
================================================================================
========== #
# Classes
#
================================================================================
========== #
*/

// Class Core.Object
// 0x0060
class UObject
{
public:
struct FPointer                                    VfTableObject;                 // 0x0000 (0x0008)
[0x0000000000821002] (CPF_Const | CPF_Native | CPF_EditConst | CPF_NoExport)
```

```
struct FPointer                                HashNext;                                // 0x0008 (0x0008)
[0x0000000000021002] (CPF_Const | CPF_Native | CPF_EditConst)
uint64_t                                       ObjectFlags;                             // 0x0010 (0x0008)
[0x0000000000021002] (CPF_Const | CPF_Native | CPF_EditConst)
struct FPointer                                HashOuterNext;                           // 0x0018 (0x0008)
[0x0000000000021002] (CPF_Const | CPF_Native | CPF_EditConst)
struct FPointer                                StateFrame;                              // 0x0020 (0x0008)
[0x0000000000021002] (CPF_Const | CPF_Native | CPF_EditConst)
class UObject*                                 Linker;                                  // 0x0028 (0x0008)
[0x0000000000821002] (CPF_Const | CPF_Native | CPF_EditConst | CPF_NoExport)
struct FPointer                                LinkerIndex;                             // 0x0030 (0x0008)
[0x0000000000821002] (CPF_Const | CPF_Native | CPF_EditConst | CPF_NoExport)
int32_t                                        ObjectInternalInteger;                   // 0x0038 (0x0004)
[0x0000000000821002] (CPF_Const | CPF_Native | CPF_EditConst | CPF_NoExport)
int32_t                                        NetIndex;                                // 0x003C (0x0004)
[0x0000000000821002] (CPF_Const | CPF_Native | CPF_EditConst | CPF_NoExport)
class UObject*                                 Outer;                                   // 0x0040 (0x0008)
[0x0000000000021002] (CPF_Const | CPF_Native | CPF_EditConst)
struct FName                                   Name;                                    // 0x0048 (0x0008)
[0x0000000000021003] (CPF_Edit | CPF_Const | CPF_Native | CPF_EditConst)
class UClass*                                  Class;                                   // 0x0050 (0x0008)
[0x0000000000021002] (CPF_Const | CPF_Native | CPF_EditConst)
class UObject*                                 ObjectArchetype;                         // 0x0058 (0x0008)
[0x0000000000021003] (CPF_Edit | CPF_Const | CPF_Native | CPF_EditConst)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Object");
}

return uClassPointer;
};

static TArray<class UObject*>* GObjObjects();

std::string GetName();
std::string GetNameCPP();
std::string GetFullName();
class UObject* GetPackageObj();
template<typename T> static T* FindObject(const std::string& objectFullName)
{
for (UObject* uObject : *UObject::GObjObjects())
{
if (uObject && uObject->IsA(T::StaticClass()))
{
if (uObject->GetFullName() == objectFullName)
{
return static_cast<T*>(uObject);
}
```

```
}
}

return nullptr;
}
template<typename T> static uint32_t CountObject(const std::string& objectName)
{
static std::map<std::string, int32_t> countCache;

if (countCache.find(objectName) == countCache.end())
{
countCache[objectName] = 0;

for (UObject* uObject : *UObject::GObjObjects())
{
if (uObject && uObject->IsA(T::StaticClass()))
{
if (uObject->GetName() == objectName)
{
countCache[uObject->GetName()]++;
}
}
}
}

return countCache[objectName];
}
static class UClass* FindClass(const std::string& classFullName);
bool IsA(class UClass* uClass);
bool IsA(int32_t objInternalInteger);

static struct FRotator RSmoothInterpTo(struct FRotator From, struct FRotator To, float Lambda,
float DeltaTime);
static struct FVector VSmoothInterpTo(struct FVector From, struct FVector To, float Lambda,
float DeltaTime);
static float FSmoothInterpTo(float From, float To, float Lambda, float DeltaTime);
static float GetSmoothInterpLerpValue(float Lambda, float DeltaTime);
static class UObject* GetTypedOuter(class UClass* ObjClass);
void MarkPendingKill();
static bool NotNone(class UObject* O);
static void SwapArrayItems();
static void Swap();
static float SumFloat(float Total, float Value);
static int32_t SumInt(int32_t Total, int32_t Value);
static int32_t SortDescendingFloat(float A, float B);
static int32_t SortAscendingFloat(float A, float B);
static int32_t SortDescendingString(class FString A, class FString B);
static int32_t SortAscendingString(class FString A, class FString B);
static int32_t SortDescendingQWORD(uint64_t A, uint64_t B);
static int32_t SortAscendingQWORD(uint64_t A, uint64_t B);
static int32_t SortDescendingInt(int32_t A, int32_t B);
static int32_t SortAscendingInt(int32_t A, int32_t B);
static class FString PadString(class FString Str, int32_t Characters);
static uint64_t GetFrameCounter();
```

```
static float GetScaledAxisValue(float Value, float Sensitivity, float MaxSensitivity);
static class UObject* GetSingleton(class UClass* ObjClass);
static class UObjectProvider* GetObjectProviderW();
static bool IsAutomationTest();
class FString ToJson();
void SetRooted(unsigned long bRooted);
static void ProfNodeEvent(class FString EventName);
static void ProfNodeSetDepthThreshold(int32_t Depth);
static void ProfNodeSetTimeThresholdSeconds(float Threshold);
static void ProfNodeStop(int32_t AssumedTimerIndex);
static int32_t ProfNodeStart(class FString TimerName);
static class FString CreateGuidString();
static class FString GetStringFromGuid(struct FGuid& InGuid);
static struct FGuid GetGuidFromString(class FString& InGuidString);
static struct FGuid CreateGuid();
static bool IsGuidValid(struct FGuid& InGuid);
static void InvalidateGuid(struct FGuid& InGuid);
class UObject* FindStructProperty(class UClass* PropertyClass, struct FName PropertyName,
struct FName StructName);
class UObject* FindProperty(class UClass* PropertyClass, struct FName PropertyName);
class UObject* DuplicateObject(class UObject* Template, class UObject* ObjOuter, class UClass*
DestClass);
float RunningAverage(float OldAverage, float NewValue, int32_t NewCount);
static float GetCurrentTimeW();
struct FLinearColor GetMaxLinearColorBrightness(struct FLinearColor C);
struct FColor GetMaxColorBrightness(struct FColor C);
static struct FLinearColor LABtoRGB(struct FLinearColor C);
static struct FLinearColor RGBtoLAB(struct FLinearColor C);
static struct FLinearColor HSVtoRGB(struct FLinearColor C);
static struct FLinearColor RGBtoHSV(struct FLinearColor C);
static struct FLinearColor IntToLinearColor(int32_t I);
static struct FColor IntToColor(int32_t I);
static int32_t LinearColorToInt(struct FLinearColor C);
static int32_t ColorToInt(struct FColor C);
bool SolveVelocityQuadratic(float Distance, float Speed, float Accel, float& Time);
float Sign(float X);
static struct FVector2D MakeVector2D(float X, float Y);
static struct FVector VAbs(struct FVector V);
static struct FVector MakeVector(float X, float Y, float Z);
struct FVector FlattenVector(struct FVector NormalToFlatten, struct FVector PlaneNormal);
struct FName GetRealArchetypeName();
static class FString FormatTime(int32_t Seconds);
static class UObject* GetTextArchetype(class UClass* ArchetypeClass, class FString Path);
bool IsArchetype();
void UnsubscribeFromAllEvents();
class UObject* NewInstance(class UObject* ObjOuter, struct FName ObjName);
void PrintDebugInfo(class UDebugDrawer* Drawer);
static struct FRotator RotatorFromInt(int32_t RotationPitchAndYaw);
static int32_t RotatorToInt(struct FRotator Rotation);
static class FString GetLanguage();
int32_t GetRandomOptionSumFrequency(TArray<float>& FreqList);
int32_t GetBuildChangelistNumber();
int32_t GetEngineVersion();
float GetAppSeconds();
```

```
void GetSystemTime(int32_t& Year, int32_t& Month, int32_t& DayOfWeek, int32_t& Day, int32_t& Hour, int32_t& Min, int32_t& Sec, int32_t& MSec);
class FString TimeStamp();
struct FVector TransformVectorByRotation(struct FRotator SourceRotation, struct FVector SourceVector, unsigned long bInverse);
struct FName GetPackageName();
bool IsPendingKill();
float RangeByteToFloatUnsigned(uint8_t inputByte);
float RangeByteToFloatSigned(uint8_t inputByte);
uint8_t FloatToRangeByteUnsigned(float inputFloat);
uint8_t FloatToRangeByteSigned(float inputFloat);
static float UnwindHeading(float A);
static float FindDeltaAngle(float A1, float A2);
static float GetHeadingAngle(struct FVector Dir);
static void GetAngularDegreesFromRadians(struct FVector2D& OutFOV);
static void GetAngularFromDotDist(struct FVector2D DotDist, struct FVector2D& OutAngDist);
static bool GetAngularDistance(struct FVector Direction, struct FVector AxisX, struct FVector AxisY, struct FVector AxisZ, struct FVector2D& OutAngularDist);
static bool GetDotDistance(struct FVector Direction, struct FVector AxisX, struct FVector AxisY, struct FVector AxisZ, struct FVector2D& OutDotDist);
bool LinePlaneIntersection(struct FVector LineStart, struct FVector LineEnd, struct FVector PlaneOrigin, struct FVector PlaneNormal, unsigned long bCheckLineSegment, struct FVector& Out_Intersection, float& Out_T);
static struct FVector PointProjectToPlane(struct FVector Point, struct FVector A, struct FVector B, struct FVector C);
float PointDistToPlane(struct FVector Point, struct FRotator Orientation, struct FVector Origin, struct FVector& out_ClosestPoint);
float PointDistToSegment(struct FVector Point, struct FVector StartPoint, struct FVector EndPoint, struct FVector& OutClosestPoint);
static float PointDistToLine(struct FVector Point, struct FVector Line, struct FVector Origin, struct FVector& OutClosestPoint);
static void GetPerObjectConfigObjects(class UClass* SearchClass, class UObject* ObjectOuter, int32_t MaxResults, class UObject*& OutObject);
static bool GetPerObjectConfigSections(class UClass* SearchClass, class UObject* ObjectOuter, int32_t MaxResults, TArray<class FString>& out_SectionNames);
static void ImportJSON(class FString PropertyName, class FString& JSON);
static void StaticSaveConfig();
void SaveConfig();
static class UObject* LoadSeekFreeObject(class UClass* ObjClass, class FString Path);
static class UObject* FindObject(class FString ObjectName, class UClass* ObjectClass);
static class UObject* DynamicLoadObject(class FString ObjectName, class UClass* ObjectClass, unsigned long MayFail);
static int32_t EnumFromString(class UObject* E, struct FName ValueName);
static struct FName GetEnum(class UObject* E, int32_t I);
void Disable(struct FName ProbeFunc);
void Enable(struct FName ProbeFunc);
void eventContinuedState();
void eventPausedState();
void eventPoppedState();
void eventPushedState();
void eventEndState(struct FName NextStateName);
void eventBeginState(struct FName PreviousStateName);
void DumpStateStack();
void PopState(unsigned long bPopAll);
```

void PushState(struct FName NewState, struct FName NewLabel);
struct FName GetStateName();
bool IsChildState(struct FName TestState, struct FName TestParentState);
bool IsInState(struct FName TestState, unsigned long bTestStateStack);
void GotoState(struct FName NewState, struct FName Label, unsigned long bForceEvents, unsigned long bKeepStack);
static bool IsUTracing();
static void SetUTracing(unsigned long bShouldUTrace);
static struct FName GetFuncName();
static void DebugBreak(int32_t UserFlags, uint8_t DebuggerType);
static class FString GetScriptTrace();
static void ScriptTrace();
static class FString ParseLocalizedPropertyPath(class FString PathName);
static class FString Localize(class FString SectionName, class FString KeyName, class FString PackageName, unsigned long bOptional);
static void WarnInternal(class FString S);
static void LogInternal(class FString S, struct FName Tag, unsigned long bFileOnly);
static struct FLinearColor LinearColorLerp(struct FLinearColor ColorA, struct FLinearColor ColorB, float Alpha);
static struct FLinearColor Subtract_LinearColorLinearColor(struct FLinearColor A, struct FLinearColor B);
static struct FLinearColor Multiply_LinearColorFloat(struct FLinearColor LC, float Mult);
static struct FLinearColor ConvertFromSRGB(struct FLinearColor OldColor);
static struct FColor LinearColorToColor(struct FLinearColor OldColor);
static struct FLinearColor ColorToLinearColor(struct FColor OldColor);
static struct FLinearColor MakeLinearColor(float R, float G, float B, float A);
static struct FColor LerpColor(struct FColor A, struct FColor B, float Alpha);
static struct FColor MakeColor(uint8_t R, uint8_t G, uint8_t B, uint8_t A);
static struct FColor Add_ColorColor(struct FColor A, struct FColor B);
static struct FColor Multiply_ColorFloat(struct FColor A, float B);
static struct FColor Multiply_FloatColor(float A, struct FColor B);
static struct FColor Subtract_ColorColor(struct FColor A, struct FColor B);
static struct FVector2D EvalInterpCurveVector2D(float InVal, struct FInterpCurveVector2D& Vector2DCurve);
static void AutoSetTangentsVector(struct FInterpCurveVector& Curve);
static struct FVector EvalInterpCurveVector(float InVal, struct FInterpCurveVector& VectorCurve);
static void AutoSetTangentsFloat(struct FInterpCurveFloat& Curve);
static float EvalInterpCurveFloat(float InVal, struct FInterpCurveFloat& FloatCurve);
static struct FVector2D vect2d(float InX, float InY);
static float GetMappedRangeValue(struct FVector2D InputRange, struct FVector2D OutputRange, float Value);
static float GetRangePctByValue(struct FVector2D Range, float Value);
static float GetRangeValueByPct(struct FVector2D Range, float Pct);
static struct FVector2D V2DNormal(struct FVector2D A);
static float V2DSizeSq(struct FVector2D A);
static float V2DSize(struct FVector2D A);
static float Dot_Vector2DVector2D(struct FVector2D A, struct FVector2D B);
static struct FVector2D SubtractEqual_Vector2DVector2D(struct FVector2D B, struct FVector2D& A);
static struct FVector2D AddEqual_Vector2DVector2D(struct FVector2D B, struct FVector2D& A);
static struct FVector2D DivideEqual_Vector2DFloat(float B, struct FVector2D& A);
static struct FVector2D MultiplyEqual_Vector2DFloat(float B, struct FVector2D& A);
static struct FVector2D Divide_Vector2DFloat(struct FVector2D A, float B);
static struct FVector2D Multiply_Vector2DFloat(struct FVector2D A, float B);

```
static struct FVector2D Subtract_Vector2DVector2D(struct FVector2D A, struct FVector2D B);
static struct FVector2D Add_Vector2DVector2D(struct FVector2D A, struct FVector2D B);
static struct FQuat Subtract_QuatQuat(struct FQuat A, struct FQuat B);
static struct FQuat Add_QuatQuat(struct FQuat A, struct FQuat B);
static struct FQuat QuatSlerp(struct FQuat A, struct FQuat B, float Alpha, unsigned long
bShortestPath);
static struct FRotator QuatToRotator(struct FQuat A);
static struct FQuat QuatFromRotator(struct FRotator A);
static struct FQuat QuatFromAxisAndAngle(struct FVector Axis, float Angle);
static struct FQuat QuatFindBetween(struct FVector A, struct FVector B);
static struct FVector QuatRotateVector(struct FQuat A, struct FVector B);
static struct FQuat QuatInvert(struct FQuat A);
static float QuatDot(struct FQuat A, struct FQuat B);
static struct FQuat QuatProduct(struct FQuat A, struct FQuat B);
static struct FVector MatrixGetAxis(struct FMatrix TM, uint8_t Axis);
static struct FVector MatrixGetOrigin(struct FMatrix TM);
static struct FRotator MatrixGetRotator(struct FMatrix TM);
static struct FMatrix MakeRotationMatrix(struct FRotator Rotation);
static struct FMatrix MakeRotationTranslationMatrix(struct FVector Translation, struct FRotator
Rotation);
static struct FVector InverseTransformNormal(struct FMatrix TM, struct FVector A);
static struct FVector TransformNormal(struct FMatrix TM, struct FVector A);
static struct FVector InverseTransformVector(struct FMatrix TM, struct FVector A);
static struct FVector TransformVector(struct FMatrix TM, struct FVector A);
static struct FMatrix Multiply_MatrixMatrix(struct FMatrix A, struct FMatrix B);
static bool NotEqual_NameName(struct FName A, struct FName B);
static bool EqualEqual_NameName(struct FName A, struct FName B);
bool IsA(struct FName ClassName);
static bool ClassIsChildOf(class UClass* TestClass, class UClass* ParentClass);
static bool NotEqual_InterfaceInterface(class UInterface* A, class UInterface* B);
static bool EqualEqual_InterfaceInterface(class UInterface* A, class UInterface* B);
static bool NotEqual_ObjectObject(class UObject* A, class UObject* B);
static bool EqualEqual_ObjectObject(class UObject* A, class UObject* B);
class FString GetPathName();
static class FString PathName(class UObject* CheckObject);
static TArray<class FString> SplitString(class FString Source, class FString Delimiter, unsigned
long bCullEmpty);
static void ParseStringIntoArray(class FString BaseString, class FString delim, unsigned long
bCullEmpty, TArray<class FString>& Pieces);
static bool ContainsWhitespace(class FString Text);
static class FString RepeatString(class FString InValue, int32_t Count);
static class FString JoinArrayQWord(class FString delim, unsigned long bIgnoreBlanks,
TArray<uint64_t>& QWordArray);
static class FString JoinArrayInt(class FString delim, unsigned long bIgnoreBlanks,
TArray<int32_t>& IntArray);
static class FString JoinArrayName(class FString delim, unsigned long bIgnoreBlanks,
TArray<struct FName>& NameArray);
static class FString JoinArrayStr(class FString delim, unsigned long bIgnoreBlanks, TArray<class
FString>& StringArray);
static void JoinArray(class FString delim, unsigned long bIgnoreBlanks, TArray<class FString>&
StringArray, class FString& out_Result);
static class FString GetRightMost(class FString Text);
static class FString Split(class FString Text, class FString SplitStr, unsigned long bOmitSplitStr);
static bool StartsWith(class FString Src, class FString Prefix);
```

static class FString Trim(class FString Src);
static class FString Repl(class FString Src, class FString Match, class FString With, unsigned long bCaseSensitive);
static int32_t Asc(class FString S);
static class FString Chr(int32_t I);
static class FString Locs(class FString S);
static class FString Caps(class FString S);
static class FString Right(class FString S, int32_t I);
static class FString Left(class FString S, int32_t I);
static class FString Mid(class FString S, int32_t I, int32_t J);
static int32_t InStr(class FString S, class FString T, unsigned long bSearchFromRight, unsigned long bIgnoreCase, int32_t StartPos);
static int32_t Len(class FString S);
static class FString SubtractEqual_StrStr(class FString B, class FString& A);
static class FString AtEqual_StrStr(class FString B, class FString& A);
static class FString ConcatEqual_StrStr(class FString B, class FString& A);
static bool ComplementEqual_StrStr(class FString A, class FString B);
static bool NotEqual_StrStr(class FString A, class FString B);
static bool EqualEqual_StrStr(class FString A, class FString B);
static bool GreaterEqual_StrStr(class FString A, class FString B);
static bool LessEqual_StrStr(class FString A, class FString B);
static bool Greater_StrStr(class FString A, class FString B);
static bool Less_StrStr(class FString A, class FString B);
static class FString At_StrStr(class FString A, class FString B);
static class FString Concat_StrStr(class FString A, class FString B);
static struct FRotator RotateRotator(struct FVector Axis, struct FRotator Rot, struct FRotator Direction, float Amount);
static struct FQuat MakeQuat(float X, float Y, float Z, float W);
static struct FRotator MakeRotator(int32_t Pitch, int32_t Yaw, int32_t Roll);
static bool SClampRotAxis(float DeltaTime, int32_t ViewAxis, int32_t MaxLimit, int32_t MinLimit, float InterpolationSpeed, int32_t& out_DeltaViewAxis);
static int32_t ClampRotAxisFromRange(int32_t Current, int32_t Min, int32_t Max);
static int32_t ClampRotAxisFromBase(int32_t Current, int32_t Center, int32_t MaxDelta);
static void ClampRotAxis(int32_t ViewAxis, int32_t MaxLimit, int32_t MinLimit, int32_t& out_DeltaViewAxis);
static struct FRotator FlattenRotatorOnAxis(struct FVector AxisToRemove, struct FRotator RotToFlatten, struct FRotator RotToFlattenTo);
static float RSize(struct FRotator R);
static float RDiff(struct FRotator A, struct FRotator B);
static int32_t NormalizeRotAxis(int32_t Angle);
static struct FRotator RInterpTo(struct FRotator Current, struct FRotator Target, float DeltaTime, float InterpSpeed, unsigned long bConstantInterpSpeed);
static struct FRotator RTransform(struct FRotator R, struct FRotator RBasis);
static struct FRotator RLerp(struct FRotator A, struct FRotator B, float Alpha, unsigned long bShortestPath);
static struct FRotator Normalize(struct FRotator Rot);
static struct FRotator OrthoRotation(struct FVector X, struct FVector Y, struct FVector Z);
static struct FRotator RotRand(unsigned long bRoll);
static struct FVector GetRotatorAxis(struct FRotator A, int32_t Axis);
static void GetUnAxes(struct FRotator A, struct FVector& X, struct FVector& Y, struct FVector& Z);
static void GetAxes(struct FRotator A, struct FVector& X, struct FVector& Y, struct FVector& Z);
static bool ClockwiseFrom_IntInt(int32_t A, int32_t B);
static struct FRotator SubtractEqual_RotatorRotator(struct FRotator B, struct FRotator& A);
static struct FRotator AddEqual_RotatorRotator(struct FRotator B, struct FRotator& A);

static struct FRotator Subtract_RotatorRotator(struct FRotator A, struct FRotator B);
static struct FRotator Add_RotatorRotator(struct FRotator A, struct FRotator B);
static struct FRotator DivideEqual_RotatorFloat(float B, struct FRotator& A);
static struct FRotator MultiplyEqual_RotatorFloat(float B, struct FRotator& A);
static struct FRotator Divide_RotatorFloat(struct FRotator A, float B);
static struct FRotator Multiply_FloatRotator(float A, struct FRotator B);
static struct FRotator Multiply_RotatorFloat(struct FRotator A, float B);
static bool NotEqual_RotatorRotator(struct FRotator A, struct FRotator B);
static bool EqualEqual_RotatorRotator(struct FRotator A, struct FRotator B);
static float GetRadiansBetweenVectors(struct FVector V0, struct FVector v1);
static struct FVector VClamp(struct FVector A, struct FVector Min, struct FVector Max);
static struct FVector vect3d(float X, float Y, float Z);
bool InCylinder(struct FVector Origin, struct FRotator Dir, float Width, struct FVector A, unsigned long bIgnoreZ);
static float NoZDot(struct FVector A, struct FVector B);
static struct FVector ClampLength(struct FVector V, float MaxLength);
static struct FVector VInterpConstantTo(struct FVector Current, struct FVector Target, float DeltaTime, float InterpSpeed);
static struct FVector VInterpTo(struct FVector Current, struct FVector Target, float DeltaTime, float InterpSpeed);
void eventConstruct();
static struct FVector ProjectOnToPlane(struct FVector InVector, struct FVector InNormal, float OverBounce);
static bool IsZero(struct FVector A);
static struct FVector ProjectOnTo(struct FVector X, struct FVector Y);
static struct FVector MirrorVectorByNormal(struct FVector InVect, struct FVector InNormal);
static struct FVector VRandCone2(struct FVector Dir, float HorizontalConeHalfAngleRadians, float VerticalConeHalfAngleRadians);
static struct FVector VRandCone(struct FVector Dir, float ConeHalfAngleRadians);
static struct FVector VRand();
static struct FVector VLerp(struct FVector A, struct FVector B, float Alpha);
static struct FVector Normal2D(struct FVector A);
static struct FVector Normal(struct FVector A);
static float VSizeSq2D(struct FVector A);
static float VSizeSq(struct FVector A);
static float VSize2D(struct FVector A);
static float VSize(struct FVector A);
static struct FVector SubtractEqual_VectorVector(struct FVector B, struct FVector& A);
static struct FVector AddEqual_VectorVector(struct FVector B, struct FVector& A);
static struct FVector DivideEqual_VectorFloat(float B, struct FVector& A);
static struct FVector MultiplyEqual_VectorVector(struct FVector B, struct FVector& A);
static struct FVector MultiplyEqual_VectorFloat(float B, struct FVector& A);
static struct FVector Cross_VectorVector(struct FVector A, struct FVector B);
static float Dot_VectorVector(struct FVector A, struct FVector B);
static bool NotEqual_VectorVector(struct FVector A, struct FVector B);
static bool EqualEqual_VectorVector(struct FVector A, struct FVector B);
static struct FVector GreaterGreater_VectorRotator(struct FVector A, struct FRotator B);
static struct FVector LessLess_VectorRotator(struct FVector A, struct FRotator B);
static struct FVector Subtract_VectorVector(struct FVector A, struct FVector B);
static struct FVector Add_VectorVector(struct FVector A, struct FVector B);
static struct FVector Divide_VectorFloat(struct FVector A, float B);
static struct FVector Multiply_VectorVector(struct FVector A, struct FVector B);
static struct FVector Multiply_FloatVector(float A, struct FVector B);
static struct FVector Multiply_VectorFloat(struct FVector A, float B);

```
static struct FVector Subtract_PreVector(struct FVector A);
static float FInterpConstantTo(float Current, float Target, float DeltaTime, float InterpSpeed);
static float FInterpTo(float Current, float Target, float DeltaTime, float InterpSpeed);
static float FPctByRange(float Value, float InMin, float InMax);
static float RandSign(float Value);
static struct FVector CalculateGravityPosition(struct FVector Location, struct FVector Velocity,
float Gravity, float Time, struct FVector GravityDirection);
static float RandRange(float InMin, float InMax);
static float FInterpEaseInOut(float A, float B, float Alpha, float Exp);
static float FInterpEaseOut(float A, float B, float Alpha, float Exp);
static float FInterpEaseIn(float A, float B, float Alpha, float Exp);
static float FCubicInterp(float P0, float T0, float P1, float T1, float A);
static int32_t FloorLog2(int32_t Value);
static int32_t FCeil(float A);
static int32_t FFloor(float A);
static int32_t Round(float A);
static float Lerp(float A, float B, float Alpha);
static float FClamp(float V, float A, float B);
static float FMax(float A, float B);
static float FMin(float A, float B);
static float FRand();
static float Square(float A);
static float Sqrt(float A);
static float Loge(float A);
static float Exp(float A);
static float Atan2(float A, float B);
static float Atan(float A);
static float Tan(float A);
static float Acos(float A);
static float Cos(float A);
static float Asin(float A);
static float Sin(float A);
static float Abs(float A);
static float SubtractEqual_FloatFloat(float B, float& A);
static float AddEqual_FloatFloat(float B, float& A);
static float DivideEqual_FloatFloat(float B, float& A);
static float MultiplyEqual_FloatFloat(float B, float& A);
static bool NotEqual_FloatFloat(float A, float B);
static bool ComplementEqual_FloatFloat(float A, float B);
static bool EqualEqual_FloatFloat(float A, float B);
static bool GreaterEqual_FloatFloat(float A, float B);
static bool LessEqual_FloatFloat(float A, float B);
static bool Greater_FloatFloat(float A, float B);
static bool Less_FloatFloat(float A, float B);
static float Subtract_FloatFloat(float A, float B);
static float Add_FloatFloat(float A, float B);
static float Percent_FloatFloat(float A, float B);
static float Divide_FloatFloat(float A, float B);
static float Multiply_FloatFloat(float A, float B);
static float MultiplyMultiply_FloatFloat(float Base, float Exp);
static float Subtract_PreFloat(float A);
static class FString ToHex(int32_t A);
static int32_t Clamp(int32_t V, int32_t A, int32_t B);
static int32_t Max(int32_t A, int32_t B);
```

```
static int32_t Min(int32_t A, int32_t B);
static int32_t Rand(int32_t Max);
static struct FColor FromHexColor(class FString Hex);
static int32_t FromHex(class FString Hex);
static uint64_t QMin(uint64_t A, uint64_t B);
static uint64_t QMax(uint64_t A, uint64_t B);
static uint64_t QSubtract(uint64_t A, uint64_t B);
static bool NotEqual_QWordInt(uint64_t A, int32_t B);
static bool EqualEqual_QWordInt(uint64_t A, int32_t B);
static bool NotEqual_QWordQWord(uint64_t A, uint64_t B);
static bool EqualEqual_QWordQWord(uint64_t A, uint64_t B);
static bool GreaterEqual_QWordQWord(uint64_t A, uint64_t B);
static bool LessEqual_QWordQWord(uint64_t A, uint64_t B);
static bool Greater_QWordQWord(uint64_t A, uint64_t B);
static bool Less_QWordQWord(uint64_t A, uint64_t B);
static int32_t Subtract_QWordQWord(uint64_t A, uint64_t B);
static uint64_t Add_QWordQWord(uint64_t A, uint64_t B);
static int32_t SubtractSubtract_Int(int32_t& A);
static int32_t AddAdd_Int(int32_t& A);
static int32_t SubtractSubtract_PreInt(int32_t& A);
static int32_t AddAdd_PreInt(int32_t& A);
static int32_t SubtractEqual_IntInt(int32_t B, int32_t& A);
static int32_t AddEqual_IntInt(int32_t B, int32_t& A);
static int32_t DivideEqual_IntFloat(float B, int32_t& A);
static int32_t MultiplyEqual_IntFloat(float B, int32_t& A);
static int32_t Or_IntInt(int32_t A, int32_t B);
static int32_t Xor_IntInt(int32_t A, int32_t B);
static int32_t And_IntInt(int32_t A, int32_t B);
static bool NotEqual_IntInt(int32_t A, int32_t B);
static bool EqualEqual_IntInt(int32_t A, int32_t B);
static bool GreaterEqual_IntInt(int32_t A, int32_t B);
static bool LessEqual_IntInt(int32_t A, int32_t B);
static bool Greater_IntInt(int32_t A, int32_t B);
static bool Less_IntInt(int32_t A, int32_t B);
static int32_t GreaterGreaterGreater_IntInt(int32_t A, int32_t B);
static int32_t GreaterGreater_IntInt(int32_t A, int32_t B);
static int32_t LessLess_IntInt(int32_t A, int32_t B);
static int32_t Subtract_IntInt(int32_t A, int32_t B);
static int32_t Add_IntInt(int32_t A, int32_t B);
static int32_t Percent_IntInt(int32_t A, int32_t B);
static int32_t Divide_IntInt(int32_t A, int32_t B);
static int32_t Multiply_IntInt(int32_t A, int32_t B);
static int32_t Subtract_PreInt(int32_t A);
static int32_t Complement_PreInt(int32_t A);
static uint8_t SubtractSubtract_Byte(uint8_t& A);
static uint8_t AddAdd_Byte(uint8_t& A);
static uint8_t SubtractSubtract_PreByte(uint8_t& A);
static uint8_t AddAdd_PreByte(uint8_t& A);
static uint8_t SubtractEqual_ByteByte(uint8_t B, uint8_t& A);
static uint8_t AddEqual_ByteByte(uint8_t B, uint8_t& A);
static uint8_t DivideEqual_ByteByte(uint8_t B, uint8_t& A);
static uint8_t MultiplyEqual_ByteFloat(float B, uint8_t& A);
static uint8_t MultiplyEqual_ByteByte(uint8_t B, uint8_t& A);
static bool OrOr_BoolBool(unsigned long A, unsigned long B);
```

```
static bool XorXor_BoolBool(unsigned long A, unsigned long B);
static bool AndAnd_BoolBool(unsigned long A, unsigned long B);
static bool NotEqual_BoolBool(unsigned long A, unsigned long B);
static bool EqualEqual_BoolBool(unsigned long A, unsigned long B);
static bool Not_PreBool(unsigned long A);
void ProcessEvent(class UFunction* uFunction, void* uParams, void* uResult);
};

// Class Core.Config_ORS
// 0x0060 (0x0060 - 0x00C0)
class UConfig_ORS : public UObject
{
public:
uint8_t                                    UnknownData00[0x60];                      // 0x0060 (0x0060)
MISSED OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Config_ORS");
}

return uClassPointer;
};

};

// Class Core.ClassTupleCollection_ORS
// 0x0060 (0x0060 - 0x00C0)
class UClassTupleCollection_ORS : public UObject
{
public:
uint8_t                                    UnknownData00[0x60];                      // 0x0060 (0x0060)
MISSED OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.ClassTupleCollection_ORS");
}

return uClassPointer;
};

};
```

```
// Class Core.ClassTuple_ORS
// 0x0068 (0x0060 - 0x00C8)
class UClassTuple_ORS : public UObject
{
public:
uint8_t                                    UnknownData00[0x68];                      // 0x0060 (0x0068)
MISSED OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.ClassTuple_ORS");
}

return uClassPointer;
};

};

// Class Core.SubscriptionCollection_ORS
// 0x0020 (0x0060 - 0x0080)
class USubscriptionCollection_ORS : public UObject
{
public:
uint8_t                                    UnknownData00[0x20];                      // 0x0060 (0x0020)
MISSED OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.SubscriptionCollection_ORS");
}

return uClassPointer;
};

};

// Class Core.Group_ORS
// 0x00D0 (0x0068 - 0x0138)
class UGroup_ORS : public UObjectScriptGroup_ORS
{
public:
uint8_t                                    UnknownData00[0xD0];                      // 0x0068 (0x00D0)
MISSED OFFSET
```

```
public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Group_ORS");
}

return uClassPointer;
};

};

// Class Core.Instance_ORS
// 0x00B0 (0x0060 - 0x0110)
class UInstance_ORS : public UObject
{
public:
uint8_t                                    UnknownData00[0xB0];                      // 0x0060 (0x00B0)
MISSED OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Instance_ORS");
}

return uClassPointer;
};

};

// Class Core.Global_ORS
// 0x0018 (0x0060 - 0x0078)
class UGlobal_ORS : public UObject
{
public:
uint8_t                                    UnknownData00[0x18];                      // 0x0060 (0x0018)
MISSED OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Global_ORS");
```

```
}

return uClassPointer;
};

};

// Class Core.UTF8
// 0x0000 (0x0060 - 0x0060)
class UUTF8 : public UObject
{
public:

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.UTF8");
}

return uClassPointer;
};

static void DecodeInline(TArray<uint8_t>& Input, class FString& Output);
static class FString Decode(TArray<uint8_t>& Input);
static void EncodeInline(class FString Input, TArray<uint8_t>& Output);
static TArray<uint8_t> Encode(class FString Input);
};

// Class Core.TextBuffer
// 0x0030 (0x0060 - 0x0090)
class UTextBuffer : public UObject
{
public:
uint8_t                                    UnknownData00[0x30];                      // 0x0060 (0x0030)
MISSED OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.TextBuffer");
}

return uClassPointer;
};

};
```

```
// Class Core.Subsystem
// 0x0008 (0x0060 - 0x0068)
class USubsystem : public UObject
{
public:
struct FPointer                                VfTable_FExec;                         // 0x0060 (0x0008)
[0x0000000000801002] (CPF_Const | CPF_Native | CPF_NoExport)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Subsystem");
}

return uClassPointer;
};

};

// Class Core.System
// 0x00F8 (0x0068 - 0x0160)
class USystem : public USubsystem
{
public:
int32_t                                        StaleCacheDays;                        // 0x0068 (0x0004)
[0x0000000000004000] (CPF_Config)
int32_t                                        MaxStaleCacheSize;                     // 0x006C (0x0004)
[0x0000000000004000] (CPF_Config)
int32_t                                        MaxOverallCacheSize;                   // 0x0070 (0x0004)
[0x0000000000004000] (CPF_Config)
int32_t                                        PackageSizeSoftLimit;                  // 0x0074 (0x0004)
[0x0000000000004000] (CPF_Config)
float                                          AsyncIOBandwidthLimit;                 // 0x0078 (0x0004)
[0x0000000000004000] (CPF_Config)
class FString                                  SavePath;                              // 0x0080 (0x0010)
[0x0000000000404000] (CPF_Config | CPF_NeedCtorLink)
class FString                                  CachePath;                             // 0x0090 (0x0010)
[0x0000000000404000] (CPF_Config | CPF_NeedCtorLink)
class FString                                  CacheExt;                              // 0x00A0 (0x0010)
[0x0000000000404000] (CPF_Config | CPF_NeedCtorLink)
TArray<class FString>                          Paths;                                 // 0x00B0 (0x0010)
[0x0000000000404000] (CPF_Config | CPF_NeedCtorLink)
TArray<class FString>                          SeekFreePCPaths;                       // 0x00C0 (0x0010)
[0x0000000000404000] (CPF_Config | CPF_NeedCtorLink)
TArray<class FString>                          ScriptPaths;                           // 0x00D0 (0x0010)
[0x0000000000404000] (CPF_Config | CPF_NeedCtorLink)
TArray<class FString>                          FRScriptPaths;                         // 0x00E0 (0x0010)
[0x0000000000404000] (CPF_Config | CPF_NeedCtorLink)
TArray<class FString>                          CutdownPaths;                          // 0x00F0 (0x0010)
```

```
[0x0000000000404000] (CPF_Config | CPF_NeedCtorLink)
TArray<struct FName>                    Suppress;                       // 0x0100 (0x0010)
[0x0000000000404000] (CPF_Config | CPF_NeedCtorLink)
TArray<struct FName>                    SuppressPublic;                 // 0x0110 (0x0010)
[0x0000000000404000] (CPF_Config | CPF_NeedCtorLink)
TArray<class FString>                   Extensions;                     // 0x0120 (0x0010)
[0x0000000000404000] (CPF_Config | CPF_NeedCtorLink)
TArray<class FString>                   SeekFreePCExtensions;           // 0x0130 (0x0010)
[0x0000000000404000] (CPF_Config | CPF_NeedCtorLink)
TArray<class FString>                   LocalizationPaths;              // 0x0140 (0x0010)
[0x0000000000404000] (CPF_Config | CPF_NeedCtorLink)
class FString                           TextureFileCacheExtension;      // 0x0150 (0x0010)
[0x0000000000404000] (CPF_Config | CPF_NeedCtorLink)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.System");
}

return uClassPointer;
};

};

// Class Core.Subscription
// 0x0018 (0x0060 - 0x0078)
class USubscription : public UObject
{
public:
struct FScriptDelegate                  __SubscriberCallback__Delegate;     // 0x0060
(0x0018) [0x0000000000400000] (CPF_NeedCtorLink)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Subscription");
}

return uClassPointer;
};

static void __Subscription__TriggerAll_0x1(class USubscription* S);
static class USubscription* GetNone();
static void TriggerAll(TArray<class USubscription*>& Subscriptions);
static class USubscription* Create(struct FScriptDelegate InCallback);
```

```
void eventDispose();
void TriggerCallback();
void SetCallback(struct FScriptDelegate InCallback);
void SubscriberCallback();
};

// Class Core.PropertyChangeDispatcher
// 0x0010 (0x0060 - 0x0070)
class UPropertyChangeDispatcher : public UObject
{
public:
uint8_t                                    UnknownData00[0x10];                      // 0x0060 (0x0010)
MISSED OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.PropertyChangeDispatcher");
}

return uClassPointer;
};

};

// Class Core.PackageMap
// 0x00B8 (0x0060 - 0x0118)
class UPackageMap : public UObject
{
public:
uint8_t                                    UnknownData00[0xB8];                      // 0x0060 (0x00B8)
MISSED OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.PackageMap");
}

return uClassPointer;
};

};

// Class Core.ObjectUtil
// 0x0000 (0x0060 - 0x0060)
```

```
class UObjectUtil : public UObject
{
public:

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.ObjectUtil");
}

return uClassPointer;
};

static class UClass* FindClass(struct FName ClassName);
static void ClearNaNValues(class UObject* InObject);
static bool IdenticalDeep(class UObject* Left, class UObject* Right);
static bool Identical(class UObject* Left, class UObject* Right);
static void InitProperties(class UObject* InObject);
static void AllCDOs(class UClass* BaseClass, unsigned long bIncludeAbstract, class UObject*&
OutCDO);
static class UObject* GetCDO(class UClass* InClass);
};

// Class Core.ObjectSerializer
// 0x0010 (0x0060 - 0x0070)
class UObjectSerializer : public UObject
{
public:
uint8_t                                    UnknownData00[0x10];                      // 0x0060 (0x0010)
MISSED OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.ObjectSerializer");
}

return uClassPointer;
};

};

// Class Core.ObjectRedirector
// 0x0008 (0x0060 - 0x0068)
class UObjectRedirector : public UObject
{
```

```
public:
uint8_t                                    UnknownData00[0x8];                       // 0x0060 (0x0008) MISSED
OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.ObjectRedirector");
}

return uClassPointer;
};

};

// Class Core.MetaData
// 0x0050 (0x0060 - 0x00B0)
class UMetaData : public UObject
{
public:
uint8_t                                    UnknownData00[0x50];                       // 0x0060 (0x0050)
MISSED OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.MetaData");
}

return uClassPointer;
};

};

// Class Core.Linker
// 0x0188 (0x0060 - 0x01E8)
class ULinker : public UObject
{
public:
uint8_t                                    UnknownData00[0x188];                       // 0x0060 (0x0188)
MISSED OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;
```

```
if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Linker");
}

return uClassPointer;
};

};

// Class Core.LinkerSave
// 0x00C0 (0x01E8 - 0x02A8)
class ULinkerSave : public ULinker
{
public:
uint8_t                                    UnknownData00[0xC0];                      // 0x01E8 (0x00C0)
MISSED OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.LinkerSave");
}

return uClassPointer;
};

};

// Class Core.LinkerLoad
// 0x0628 (0x01E8 - 0x0810)
class ULinkerLoad : public ULinker
{
public:
uint8_t                                    UnknownData00[0x628];                      // 0x01E8 (0x0628)
MISSED OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.LinkerLoad");
}

return uClassPointer;
};
```

```
};

// Class Core.Interface
// 0x0000 (0x0060 - 0x0060)
class UInterface : public UObject
{
public:

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Interface");
}

return uClassPointer;
};

};

// Class Core.FileSystem
// 0x0000 (0x0060 - 0x0060)
class UFileSystem : public UObject
{
public:

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.FileSystem");
}

return uClassPointer;
};

static bool IsCookedBuild();
static void CloseLogFile();
static class FString GetLogFileName();
static bool DeleteDirectoryTree(class FString Path);
static bool DeleteFileW(class FString Path);
static bool AppendStringToFile(class FString Path, class FString Text);
static bool SaveStringToFile(class FString Path, class FString Text);
static bool SaveBytesToFile(class FString Path, TArray<uint8_t>& Bytes);
static bool LoadFileToBytes(class FString Path, int32_t StartOffset, int32_t Length,
TArray<uint8_t>& OutBytes);
static bool LoadFileToString(class FString Path, class FString& OutText);
```

```
static int32_t GetFileSize(class FString Path);
static class FString GetFileExtensionWithoutDot(class FString Path);
static class FString GetFileExtension(class FString Path);
static class FString GetFilePathWithoutExtension(class FString Path);
static class FString GetFileNameWithoutExtension(class FString Path);
static class FString GetFilename(class FString Path);
static void FindFiles(class FString Path, TArray<class FString>& OutFilenames);
};

// Class Core.Field
// 0x0010 (0x0060 - 0x0070)
class UField : public UObject
{
public:
class UField* Next; // 0x0060 (0x0008)
uint8_t UnknownData00[0x8]; // 0x0068 (0x0008) DYNAMIC FIELD PADDING

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Field");
}

return uClassPointer;
};

};

// Class Core.Struct
// 0x00C0 (0x0070 - 0x0130)
class UStruct : public UField
{
public:
uint8_t UnknownData00[0x10]; // 0x0070 (0x0010) DYNAMIC FIELD PADDING
class UField* SuperField; // 0x0080 (0x0008)
class UField* Children; // 0x0088 (0x0008)
unsigned long PropertySize; // 0x0090 (0x0004)
uint8_t UnknownData01[0x9C]; // 0x0094 (0x009C) DYNAMIC FIELD PADDING

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Struct");
}

return uClassPointer;
```

```
};

};

// Class Core.ScriptStruct
// 0x0028 (0x0130 - 0x0158)
class UScriptStruct : public UStruct
{
public:
uint8_t                                    UnknownData00[0x28];                      // 0x0130 (0x0028) MISSED OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.ScriptStruct");
}

return uClassPointer;
};

};

// Class Core.Function
// 0x0030 (0x0130 - 0x0160)
class UFunction : public UStruct
{
public:
uint64_t FunctionFlags; // 0x0130 (0x0008)
uint16_t iNative; // 0x0138 (0x0002)
uint8_t UnknownData00[0x26]; // 0x013A (0x0026) DYNAMIC FIELD PADDING

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Function");
}

return uClassPointer;
};

static UFunction* FindFunction(const std::string& functionFullName);
};

// Class Core.Property
// 0x0058 (0x0070 - 0x00C8)
```

```
class UProperty : public UField
{
public:
unsigned long ArrayDim; // 0x0070 (0x0004)
unsigned long ElementSize; // 0x0074 (0x0004)
uint64_t PropertyFlags; // 0x0078 (0x0008)
uint8_t UnknownData00[0x18]; // 0x0080 (0x0018) DYNAMIC FIELD PADDING
unsigned long Offset; // 0x0098 (0x0004)
uint8_t UnknownData01[0x2C]; // 0x009C (0x002C) DYNAMIC FIELD PADDING

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Property");
}

return uClassPointer;
};

};

// Class Core.StructProperty
// 0x0008 (0x00C8 - 0x00D0)
class UStructProperty : public UProperty
{
public:
class UStruct* Struct; // 0x00C8 (0x0008)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.StructProperty");
}

return uClassPointer;
};

};

// Class Core.StrProperty
// 0x0000 (0x00C8 - 0x00C8)
class UStrProperty : public UProperty
{
public:

public:
```

```
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.StrProperty");
}

return uClassPointer;
};

};

// Class Core.QWordProperty
// 0x0000 (0x00C8 - 0x00C8)
class UQWordProperty : public UProperty
{
public:

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.QWordProperty");
}

return uClassPointer;
};

};

// Class Core.ObjectProperty
// 0x0010 (0x00C8 - 0x00D8)
class UObjectProperty : public UProperty
{
public:
class UClass* PropertyClass; // 0x00C8 (0x0008)
uint8_t UnknownData00[0x8]; // 0x00D0 (0x0008) DYNAMIC FIELD PADDING

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.ObjectProperty");
}

return uClassPointer;
```

```
};

};

// Class Core.ComponentProperty
// 0x0000 (0x00D8 - 0x00D8)
class UComponentProperty : public UObjectProperty
{
public:

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.ComponentProperty");
}

return uClassPointer;
};

};

// Class Core.ClassProperty
// 0x0008 (0x00D8 - 0x00E0)
class UClassProperty : public UObjectProperty
{
public:
uint8_t                                    UnknownData00[0x8];                       // 0x00D8 (0x0008) MISSED
OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.ClassProperty");
}

return uClassPointer;
};

};

// Class Core.NameProperty
// 0x0000 (0x00C8 - 0x00C8)
class UNameProperty : public UProperty
{
public:
```

```
public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.NameProperty");
}

return uClassPointer;
};

};

// Class Core.MapProperty
// 0x0010 (0x00C8 - 0x00D8)
class UMapProperty : public UProperty
{
public:
class UProperty* Key; // 0x00C8 (0x0008)
class UProperty* Value; // 0x00D0 (0x0008)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.MapProperty");
}

return uClassPointer;
};

};

// Class Core.IntProperty
// 0x0000 (0x00C8 - 0x00C8)
class UIntProperty : public UProperty
{
public:

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.IntProperty");
}
```

```
return uClassPointer;
};

};

// Class Core.InterfaceProperty
// 0x0010 (0x00C8 - 0x00D8)
class UInterfaceProperty : public UProperty
{
public:
class UClass* InterfaceClass; // 0x00C8 (0x0008)
uint8_t UnknownData00[0x8]; // 0x00D0 (0x0008) DYNAMIC FIELD PADDING

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.InterfaceProperty");
}

return uClassPointer;
};

};

// Class Core.FloatProperty
// 0x0000 (0x00C8 - 0x00C8)
class UFloatProperty : public UProperty
{
public:

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.FloatProperty");
}

return uClassPointer;
};

};

// Class Core.DelegateProperty
// 0x0010 (0x00C8 - 0x00D8)
class UDelegateProperty : public UProperty
{
public:
```

```
class UFuncton* Function; // 0x00C8 (0x0008)
struct FName Name; // 0x00D0 (0x0008)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.DelegateProperty");
}

return uClassPointer;
};

};

// Class Core.ByteProperty
// 0x0008 (0x00C8 - 0x00D0)
class UByteProperty : public UProperty
{
public:
class UEnum* Enum; // 0x00C8 (0x0008)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.ByteProperty");
}

return uClassPointer;
};

};

// Class Core.BoolProperty
// 0x0008 (0x00C8 - 0x00D0)
class UBoolProperty : public UProperty
{
public:
uint64_t BitMask; // 0x00C8 (0x0008)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
```

```
uClassPointer = UObject::FindClass("Class Core.BoolProperty");
}

return uClassPointer;
};

};

// Class Core.ArrayProperty
// 0x0008 (0x00C8 - 0x00D0)
class UArrayProperty : public UProperty
{
public:
class UProperty* Inner; // 0x00C8 (0x0008)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.ArrayProperty");
}

return uClassPointer;
};

};

// Class Core.Enum
// 0x0010 (0x0070 - 0x0080)
class UEnum : public UField
{
public:
TArray<struct FName> Names; // 0x0070 (0x0010)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Enum");
}

return uClassPointer;
};

};

// Class Core.Const
// 0x0010 (0x0070 - 0x0080)
```

```
class UConst : public UField
{
public:
class FString Value; // 0x0070 (0x0010)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Const");
}

return uClassPointer;
};

};

// Class Core.FeatureSystem
// 0x0020 (0x0060 - 0x0080)
class UFeatureSystem : public UObject
{
public:
uint8_t                                    UnknownData00[0x8];                       // 0x0060 (0x0008) MISSED
OFFSET
unsigned long                              Prime : 1;                                // 0x0068 (0x0004)
[0x0000000040000000] [0x00000002] (CPF_EditInlineNotify)
unsigned long                              Matchmaking : 1;                          // 0x0068 (0x0004)
[0x0000000040000000] [0x00000008] (CPF_EditInlineNotify)
unsigned long                              PrivateMatch : 1;                         // 0x0068 (0x0004)
[0x0000000040000000] [0x00000010] (CPF_EditInlineNotify)
unsigned long                              SplitscreenMatch : 1;                     // 0x0068 (0x0004)
[0x0000000040000000] [0x00000020] (CPF_EditInlineNotify)
unsigned long                              SplitscreenJoin : 1;                      // 0x0068 (0x0004)
[0x0000000040000000] [0x00000040] (CPF_EditInlineNotify)
unsigned long                              SeasonMode : 1;                           // 0x0068 (0x0004)
[0x0000000040000000] [0x00000080] (CPF_EditInlineNotify)
unsigned long                              Tutorial : 1;                             // 0x0068 (0x0004)
[0x0000000040000000] [0x00000100] (CPF_EditInlineNotify)
unsigned long                              Garage : 1;                               // 0x0068 (0x0004)
[0x0000000040000000] [0x00000200] (CPF_EditInlineNotify)
unsigned long                              Options : 1;                              // 0x0068 (0x0004)
[0x0000000040000000] [0x00000400] (CPF_EditInlineNotify)
unsigned long                              ReplaySaves : 1;                          // 0x0068 (0x0004)
[0x0000000040000000] [0x00000800] (CPF_EditInlineNotify)
unsigned long                              MainMenu : 1;                             // 0x0068 (0x0004)
[0x0000000040000000] [0x00001000] (CPF_EditInlineNotify)
unsigned long                              MidgameMenu : 1;                          // 0x0068 (0x0004)
[0x0000000040000000] [0x00002000] (CPF_EditInlineNotify)
unsigned long                              Party : 1;                                // 0x0068 (0x0004)
[0x0000000040000000] [0x00004000] (CPF_EditInlineNotify)
unsigned long                              PsyNetParty : 1;                          // 0x0068 (0x0004)
```

[0x0000000040000000] [0x00008000] (CPF_EditInlineNotify)
unsigned long Achievements : 1; // 0x0068 (0x0004)
[0x0000000040000000] [0x00010000] (CPF_EditInlineNotify)
unsigned long Stats : 1; // 0x0068 (0x0004)
[0x0000000040000000] [0x00020000] (CPF_EditInlineNotify)
unsigned long Leaderboards : 1; // 0x0068 (0x0004)
[0x0000000040000000] [0x00040000] (CPF_EditInlineNotify)
unsigned long XP : 1; // 0x0068 (0x0004)
[0x0000000040000000] [0x00080000] (CPF_EditInlineNotify)
unsigned long Chat : 1; // 0x0068 (0x0004)
[0x0000000040000000] [0x00100000] (CPF_EditInlineNotify)
unsigned long TrainingDifficulties : 1; // 0x0068 (0x0004)
[0x0000000040000000] [0x00200000] (CPF_EditInlineNotify)
unsigned long Spectator : 1; // 0x0068 (0x0004)
[0x0000000040000000] [0x00400000] (CPF_EditInlineNotify)
unsigned long CrossPlatformPrivateMatch : 1; // 0x0068 (0x0004)
[0x0000000040000000] [0x00800000] (CPF_EditInlineNotify)
unsigned long Lan : 1; // 0x0068 (0x0004)
[0x0000000040000000] [0x01000000] (CPF_EditInlineNotify)
unsigned long PlayerReporting : 1; // 0x0068 (0x0004)
[0x0000000040000000] [0x02000000] (CPF_EditInlineNotify)
unsigned long OnlineServices : 1; // 0x0068 (0x0004)
[0x0000000040000000] [0x40000000] (CPF_EditInlineNotify)
unsigned long RemoveCrossPlatformProducts : 1; // 0x0068
(0x0004) [0x0000000040000000] [0x80000000] (CPF_EditInlineNotify)
unsigned long ProductValidation : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x00000001] (CPF_EditInlineNotify)
unsigned long MapPrefs : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x00000002] (CPF_EditInlineNotify)
unsigned long Tournaments : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x00000008] (CPF_EditInlineNotify)
unsigned long PreMatchLobby : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x00000010] (CPF_EditInlineNotify)
unsigned long Challenges : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x00000020] (CPF_EditInlineNotify)
unsigned long AntiAddiction : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x00000040] (CPF_EditInlineNotify)
unsigned long TrainingEditor : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x00000080] (CPF_EditInlineNotify)
unsigned long VoiceChat : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x00000100] (CPF_EditInlineNotify)
unsigned long SplitScreen : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x00000200] (CPF_EditInlineNotify)
unsigned long Clubs : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x00000400] (CPF_EditInlineNotify)
unsigned long FilterContent : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x00000800] (CPF_EditInlineNotify)
unsigned long EncryptContent : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x00001000] (CPF_EditInlineNotify)
unsigned long EsportsCamera : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x00008000] (CPF_EditInlineNotify)
unsigned long OnlineXP : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x00010000] (CPF_EditInlineNotify)
unsigned long ClanforgeReservation : 1; // 0x006C (0x0004)

[0x0000000040000000] [0x00040000] (CPF_EditInlineNotify)
unsigned long UserSettingObserver : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x00080000] (CPF_EditInlineNotify)
unsigned long Metrics : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x00100000] (CPF_EditInlineNotify)
unsigned long EOSMetrics : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x00200000] (CPF_EditInlineNotify)
unsigned long MusicPlaylistSelection : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x00400000] (CPF_EditInlineNotify)
unsigned long SpecialEvents : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x00800000] (CPF_EditInlineNotify)
unsigned long OnlineShop : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x01000000] (CPF_EditInlineNotify)
unsigned long PlayerBannerCustomization : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x021000000] (CPF_EditInlineNotify)
unsigned long SecureUDP : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x042000000] (CPF_EditInlineNotify)
unsigned long PsyNet : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x084000000] (CPF_EditInlineNotify)
unsigned long OnlinePlayerStorage : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x108000000] (CPF_EditInlineNotify)
unsigned long LocalSaveData : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x210000000] (CPF_EditInlineNotify)
unsigned long CrowdV2 : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x420000000] (CPF_EditInlineNotify)
unsigned long ChatBan : 1; // 0x006C (0x0004)
[0x0000000040000000] [0x840000000] (CPF_EditInlineNotify)
unsigned long BacktraceCrashDumps : 1; // 0x00706C (0x0004)
[0x0000000040000000] [0x800000001] (CPF_EditInlineNotify)
unsigned long SpotifyButton : 1; // 0x0070 (0x0004)
[0x0000000040000000] [0x000000021] (CPF_EditInlineNotify)
unsigned long RocketPass : 1; // 0x0070 (0x0004)
[0x0000000040000000] [0x000000042] (CPF_EditInlineNotify)
unsigned long SeasonRewards : 1; // 0x0070 (0x0004)
[0x0000000040000000] [0x000000108] (CPF_EditInlineNotify)
unsigned long FaceIt : 1; // 0x0070 (0x0004)
[0x0000000040000000] [0x000000210] (CPF_EditInlineNotify)
unsigned long KnockOut : 1; // 0x0070 (0x0004)
[0x0000000040000000] [0x000000420] (CPF_EditInlineNotify)
unsigned long OnlinePlayerTitles : 1; // 0x0070 (0x0004)
[0x0000000040000000] [0x000000840] (CPF_EditInlineNotify)
unsigned long RestrictByRegion : 1; // 0x0070 (0x0004)
[0x0000000040000000] [0x000001080] (CPF_EditInlineNotify)
unsigned long FirstTimeExperience : 1; // 0x0070 (0x0004)
[0x0000000040000000] [0x000004200] (CPF_EditInlineNotify)
unsigned long RLBot : 1; // 0x0070 (0x0004)
[0x0000000040000000] [0x000008400] (CPF_EditInlineNotify)
unsigned long UserBugReport : 1; // 0x0070 (0x0004)
[0x0000000040000000] [0x000010800] (CPF_EditInlineNotify)
unsigned long SteamInput : 1; // 0x0070 (0x0004)
[0x0000000040000000] [0x000021000] (CPF_EditInlineNotify)
unsigned long ReplayFXControls : 1; // 0x0070 (0x0004)
[0x0000000040000000] [0x000420000] (CPF_EditInlineNotify)
unsigned long ESportsShop : 1; // 0x0070 (0x0004)

[0x0000000040000000] [0x000840000] (CPF_EditInlineNotify)
unsigned long DynamicRangeAudioSettings : 1; // 0x0070 (0x0004)
[0x0000000040000000] [0x008400000] (CPF_EditInlineNotify)
unsigned long AutoTour : 1; // 0x0070 (0x0004)
[0x0000000040000000] [0x010800000] (CPF_EditInlineNotify)
unsigned long QuickPlay : 1; // 0x0070 (0x0004)
[0x0000000040000000] [0x021000000] (CPF_EditInlineNotify)
unsigned long NewsPanelV2 : 1; // 0x0070 (0x0004)
[0x0000000040000000] [0x042000000] (CPF_EditInlineNotify)
unsigned long Blueprints : 1; // 0x0070 (0x0004)
[0x0000000040000000] [0x084000000] (CPF_EditInlineNotify)
unsigned long GodBall : 1; // 0x0070 (0x0004)
[0x0000000040000000] [0x108000000] (CPF_EditInlineNotify)
unsigned long RocketBucks : 1; // 0x0070 (0x0004)
[0x0000000040000000] [0x420000000] (CPF_EditInlineNotify)
unsigned long DiscordRichPresence : 1; // 0x0074 (0x0004)
[0x0000000040000000] [0x000000042] (CPF_EditInlineNotify)
unsigned long SupportACreator : 1; // 0x0074 (0x0004)
[0x0000000040000000] [0x000000084] (CPF_EditInlineNotify)
unsigned long CinematicIntro : 1; // 0x0074 (0x0004)
[0x0000000040000000] [0x000000108] (CPF_EditInlineNotify)
unsigned long TinyCrowd : 1; // 0x0074 (0x0004)
[0x0000000040000000] [0x000000210] (CPF_EditInlineNotify)
unsigned long CrumbTrail : 1; // 0x0074 (0x0004)
[0x0000000040000000] [0x000000420] (CPF_EditInlineNotify)
unsigned long EpicGameStoreBuild : 1; // 0x0074 (0x0004)
[0x0000000040000000] [0x000000840] (CPF_EditInlineNotify)
unsigned long XPGatedPlaylists : 1; // 0x0074 (0x0004)
[0x0000000040000000] [0x000001080] (CPF_EditInlineNotify)
unsigned long TradeInV2 : 1; // 0x0074 (0x0004)
[0x0000000040000000] [0x000004200] (CPF_EditInlineNotify)
unsigned long Football : 1; // 0x0074 (0x0004)
[0x0000000040000000] [0x000008400] (CPF_EditInlineNotify)
unsigned long RumbleSelection : 1; // 0x0074 (0x0004)
[0x0000000040000000] [0x000010800] (CPF_EditInlineNotify)
unsigned long UndersizedParty : 1; // 0x0074 (0x0004)
[0x0000000040000000] [0x000021000] (CPF_EditInlineNotify)
unsigned long StreamerSafeAudio : 1; // 0x0074 (0x0004)
[0x0000000040000000] [0x000042000] (CPF_EditInlineNotify)
unsigned long FreeplayCommands : 1; // 0x0074 (0x0004)
[0x0000000040000000] [0x000084000] (CPF_EditInlineNotify)
unsigned long Rumble_BM : 1; // 0x0074 (0x0004)
[0x0000000040000000] [0x000108000] (CPF_EditInlineNotify)
unsigned long PlayerReportingV2 : 1; // 0x0074 (0x0004)
[0x0000000040000000] [0x000210000] (CPF_EditInlineNotify)
unsigned long BlogScheduling : 1; // 0x0074 (0x0004)
[0x0000000040000000] [0x000420000] (CPF_EditInlineNotify)
unsigned long EOSVoice : 1; // 0x0074 (0x0004)
[0x0000000040000000] [0x000840000] (CPF_EditInlineNotify)
unsigned long QuickPostMatchRequeue : 1; // 0x0074 (0x0004)
[0x0000000040000000] [0x001080000] (CPF_EditInlineNotify)
unsigned long TrainingNavigation : 1; // 0x0074 (0x0004)
[0x0000000040000000] [0x008400000] (CPF_EditInlineNotify)
unsigned long TrainingManipulation : 1; // 0x0074 (0x0004)

```
[0x0000000040000000] [0x010800000] (CPF_EditInlineNotify)
unsigned long                      FilterByColor : 1;                  // 0x0074 (0x0004)
[0x0000000040000000] [0x042000000] (CPF_EditInlineNotify)
unsigned long                      Scoreboard : 1;                     // 0x0074 (0x0004)
[0x0000000040000000] [0x084000000] (CPF_EditInlineNotify)
unsigned long                      DynamicMapEvents : 1;               // 0x0074 (0x0004)
[0x0000000040000000] [0x108000000] (CPF_EditInlineNotify)
unsigned long                      NameplateBoost : 1;                 // 0x0074 (0x0004)
[0x0000000040000000] [0x420000000] (CPF_EditInlineNotify)
unsigned long                      EOSGameClips : 1;                   // 0x0074 (0x0004)
[0x0000000040000000] [0x840000000] (CPF_EditInlineNotify)
unsigned long                      DynamicLogos : 1;                   // 0x00784 (0x0004)
[0x0000000040000000] [0x800000001] (CPF_EditInlineNotify)
unsigned long                      XETagging : 1;                      // 0x0078 (0x0004)
[0x0000000040000000] [0x000000021] (CPF_EditInlineNotify)
unsigned long                      PlayMenuV4 : 1;                     // 0x0078 (0x0004)
[0x0000000040000000] [0x000000042] (CPF_EditInlineNotify)
unsigned long                      QuickChatTimeStamp : 1;             // 0x0078 (0x0004)
[0x0000000040000000] [0x00000008] (CPF_EditInlineNotify)
unsigned long                      BundleProration : 1;                // 0x0078 (0x0004)
[0x0000000040000000] [0x00000010] (CPF_EditInlineNotify)
unsigned long                      TeamDemoAudio : 1;                  // 0x0078 (0x0004)
[0x0000000040000000] [0x00000020] (CPF_EditInlineNotify)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.FeatureSystem");
}

return uClassPointer;
};

};

// Class Core.Factory
// 0x0058 (0x0060 - 0x00B8)
class UFactory : public UObject
{
public:
class FString                      Category;                           // 0x0060 (0x0010)
[0x0000000000400000] (CPF_NeedCtorLink)
class UClass*                      SupportedClass;                     // 0x0070 (0x0008)
[0x0000000000000000]
class UClass*                      ContextClass;                       // 0x0078 (0x0008)
[0x0000000000000000]
class FString                      Description;                        // 0x0080 (0x0010)
[0x0000000000400000] (CPF_NeedCtorLink)
TArray<class FString>              Formats;                            // 0x0090 (0x0010)
[0x0000000000400000] (CPF_NeedCtorLink)
```

```
unsigned long                      bCreateNew : 1;                    // 0x00A0 (0x0004) [0x0000000000000000] [0x00000001]
unsigned long                      bEditAfterNew : 1;                 // 0x00A0 (0x0004) [0x0000000000000000] [0x00000002]
unsigned long                      bEditorImport : 1;                 // 0x00A0 (0x0004) [0x0000000000000000] [0x00000004]
unsigned long                      bText : 1;                         // 0x00A0 (0x0004) [0x0000000000000000] [0x00000008]
unsigned long                      bAssetNameMatchesPackageName : 1;  // 0x00A0 (0x0004) [0x0000000000000000] [0x00000010]
int32_t                            AutoPriority;                      // 0x00A4 (0x0004) [0x0000000000000000]
TArray<class FString>              ValidGameNames;                    // 0x00A8 (0x0010) [0x0000000000400000] (CPF_NeedCtorLink)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Factory");
}

return uClassPointer;
};

};

// Class Core.TextBufferFactory
// 0x0000 (0x00B8 - 0x00B8)
class UTextBufferFactory : public UFactory
{
public:

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.TextBufferFactory");
}

return uClassPointer;
};

};

// Class Core.Exporter
// 0x0038 (0x0060 - 0x0098)
class UExporter : public UObject
```

```
{
public:
uint8_t                                    UnknownData00[0x8];                       // 0x0060 (0x0008) MISSED
OFFSET
TArray<class FString>                      FormatExtension;                          // 0x0068 (0x0010)
[0x0000000000400000] (CPF_NeedCtorLink)
TArray<class FString>                      FormatDescription;                        // 0x0078 (0x0010)
[0x0000000000400000] (CPF_NeedCtorLink)
uint8_t                                    UnknownData01[0x10];                      // 0x0088 (0x0010)
MISSED OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Exporter");
}

return uClassPointer;
};

};

// Class Core.ErrorType
// 0x0010 (0x0060 - 0x0070)
class UErrorType : public UObject
{
public:
class FString                              LocalizationKey;                          // 0x0060 (0x0010)
[0x0000000000400002] (CPF_Const | CPF_NeedCtorLink)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.ErrorType");
}

return uClassPointer;
};

class UError* CreateError(class FString InErrorMessage, int32_t InErrorCode);
class FString GetLocalizedMessage();
};

// Class Core.ErrorList
// 0x0020 (0x0060 - 0x0080)
class UErrorList : public UObject
```

```
{
public:
class FString                                  LocalizationPackage;                       // 0x0060 (0x0010)
[0x0000000000400003] (CPF_Edit | CPF_Const | CPF_NeedCtorLink)
class FString                                  LocalizationSection;                       // 0x0070 (0x0010)
[0x0000000000400003] (CPF_Edit | CPF_Const | CPF_NeedCtorLink)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.ErrorList");
}

return uClassPointer;
};

static class UErrorType* GetErrorType(struct FName Error);
};

// Class Core.ArrayErrors
// 0x0010 (0x0080 - 0x0090)
class UArrayErrors : public UErrorList
{
public:
class UErrorType*                              Remove_NegativeNumberOfElements;           // 0x0080
(0x0008) [0x0000000000000002] (CPF_Const)
class UErrorType*                              Remove_OutOfBounds;                        // 0x0088 (0x0008)
[0x0000000000000002] (CPF_Const)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.ArrayErrors");
}

return uClassPointer;
};

};

// Class Core.Error
// 0x0024 (0x0060 - 0x0084)
class UError : public UObject
{
public:
class UErrorType*                              Type;                                      // 0x0060 (0x0008)
```

```
[0x0000000000002002] (CPF_Const | CPF_Transient)
class FString                                  Message;                                  // 0x0068 (0x0010)
[0x0000000000402002] (CPF_Const | CPF_Transient | CPF_NeedCtorLink)
int32_t                                        Code;                                     // 0x0078 (0x0004)
[0x0000000000002002] (CPF_Const | CPF_Transient)
struct FName                                   RetryKey;                                 // 0x007C (0x0008)
[0x0000000000002002] (CPF_Const | CPF_Transient)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Error");
}

return uClassPointer;
};

class FString GetDebugMessage();
class FString GetLocalizedMessage();
};

// Class Core.DelegateTracker
// 0x0028 (0x0060 - 0x0088)
class UDelegateTracker : public UObject
{
public:
TArray<struct FAsyncDelegateInfo>              AsyncDelegates;                           // 0x0060
(0x0010) [0x0000000000400000] (CPF_NeedCtorLink)
struct FScriptDelegate                         __PlaceholderDelegate__Delegate;          // 0x0070
(0x0018) [0x0000000000400000] (CPF_NeedCtorLink)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.DelegateTracker");
}

return uClassPointer;
};

struct FScriptDelegate RemoveDelegate(int32_t CallbackId);
int32_t AddDelegate(struct FScriptDelegate Callback);
void PlaceholderDelegate();
};

// Class Core.DebugDrawer
```

```
// 0x0060 (0x0060 - 0x00C0)
class UDebugDrawer : public UObject
{
public:
struct FColor                                  DefaultTextColor;                          // 0x0060 (0x0004) [0x0000000000000001] (CPF_Edit)
unsigned long                                  bSilent : 1;                               // 0x0064 (0x0004) [0x0000000000000001] [0x00000001] (CPF_Edit)
unsigned long                                  bPrintActorsInline : 1;                    // 0x0064 (0x0004) [0x0000000000000001] [0x00000002] (CPF_Edit)
int32_t                                        Indentation;                               // 0x0068 (0x0004) [0x0000008000002000] (CPF_Transient)
class FString                                  IndentationString;                         // 0x0070 (0x0010) [0x0000008000402000] (CPF_Transient | CPF_NeedCtorLink)
TArray<class UObject*>                         PrintedObjects;                            // 0x0080 (0x0010) [0x0000000000402000] (CPF_Transient | CPF_NeedCtorLink)
TArray<class UObject*>                         QueuedObjects;                             // 0x0090 (0x0010) [0x0000000000402000] (CPF_Transient | CPF_NeedCtorLink)
int32_t                                        PrintObjectCount;                          // 0x00A0 (0x0004) [0x0000000000002000] (CPF_Transient)
struct FScriptDelegate                         __LogFunc__Delegate;                       // 0x00A8 (0x0018) [0x0000000000400000] (CPF_NeedCtorLink)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.DebugDrawer");
}

return uClassPointer;
};

void Reset();
void PrintText(class FString Text, struct FColor InColor);
void eventPrintArrayProperty(class FString PropertyName, int32_t Index, class FString Value);
void eventPrintProperty(class FString PropertyName, class FString Value);
void eventEndSection();
void eventStartSection();
void eventPrintObject(class FString Title, class UObject* ForObj);
void eventPrintSeperater();
void eventDebugArrayObject(class FString Title, int32_t Index, class UObject* ForObj);
void eventDebugObject(class FString Title, class UObject* ForObj);
bool ShouldDisplayDebug(struct FName Category);
void LogFunc(class FString Str);
};

// Class Core.Compression
// 0x0000 (0x0060 - 0x0060)
class UCompression : public UObject
{
```

```
public:

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Compression");
}

return uClassPointer;
};

static bool ZLibCompress(TArray<uint8_t>& Uncompressed, TArray<uint8_t>& OutCompressed);
};

// Class Core.Component
// 0x0010 (0x0060 - 0x0070)
class UComponent : public UObject
{
public:
class UClass*                                 TemplateOwnerClass;                        // 0x0060 (0x0008)
[0x0000000000001002] (CPF_Const | CPF_Native)
struct FName                                  TemplateName;                              // 0x0068 (0x0008)
[0x0000000000001002] (CPF_Const | CPF_Native)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Component");
}

return uClassPointer;
};

};

// Class Core.StringObjectMap
// 0x0050 (0x0070 - 0x00C0)
class UStringObjectMap : public UComponent
{
public:
struct FMap_Mirror                            Map;                                       // 0x0070 (0x0050)
[0x0000000000001002] (CPF_Const | CPF_Native)

public:
static UClass* StaticClass()
{
```

```
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.StringObjectMap");
}

return uClassPointer;
};

bool Contains(class FString Key);
void Remove(class FString Key);
bool TryGetObjectW(class FString Key, class UObject*& OutValue);
void TryGet();
class UObject* GetObjectW(class FString Key);
void Get();
void Set(class FString Key, class UObject* Value);
};

// Class Core.StringMap
// 0x0068 (0x0070 - 0x00D8)
class UStringMap : public UComponent
{
public:
struct FMap_Mirror                             Map;                                       // 0x0070 (0x0050)
[0x0000000000001002] (CPF_Const | CPF_Native)
struct FScriptDelegate                         __PairCallback__Delegate;                  // 0x00C0 (0x0018)
[0x0000000000400000] (CPF_NeedCtorLink)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.StringMap");
}

return uClassPointer;
};

void ForEach(struct FScriptDelegate Callback);
void Append(class UStringMap* Other);
bool Contains(class FString Key);
void Remove(class FString Key);
bool TryGet(class FString Key, class FString& OutValue);
class FString Get(class FString Key);
void Set(class FString Key, class FString Value);
void PairCallback(class FString Key, class FString Value);
};

// Class Core.ObjectProvider
// 0x0170 (0x0070 - 0x01E0)
```

```
class UObjectProvider : public UComponent
{
public:
struct FPointer                                VfTable_FObjectDestructionSubscriber;         // 0x0070
(0x0008) [0x0000000000801002] (CPF_Const | CPF_Native | CPF_NoExport)
TArray<class UObject*>                         MyObjects;                                    // 0x0078 (0x0010)
[0x0000004000402002] (CPF_Const | CPF_Transient | CPF_NeedCtorLink)
struct FMap_Mirror                             ObjectRefs;                                   // 0x0088 (0x0050)
[0x0000000000003002] (CPF_Const | CPF_Native | CPF_Transient)
TArray<class UObject*>                         TreeObjects;                                  // 0x00D8 (0x0010)
[0x0000000000402002] (CPF_Const | CPF_Transient | CPF_NeedCtorLink)
TArray<struct FObjectProviderSubscription>     SubscribedToAdds;                             // 0x00E8
(0x0010) [0x0000000000402002] (CPF_Const | CPF_Transient | CPF_NeedCtorLink)
TArray<struct FObjectProviderSubscription>     SubscribedToRemoves;                          //
0x00F8 (0x0010) [0x0000000000402002] (CPF_Const | CPF_Transient | CPF_NeedCtorLink)
TArray<struct FObjectProviderSubscription>     SubscribedToLists;                            // 0x0108
(0x0010) [0x0000000000402002] (CPF_Const | CPF_Transient | CPF_NeedCtorLink)
struct FArray_Mirror                           Injections;                                   // 0x0118 (0x0010)
[0x0000000000003002] (CPF_Const | CPF_Native | CPF_Transient)
struct FArray_Mirror                           InterfaceInjections;                          // 0x0128 (0x0010)
[0x0000000000003002] (CPF_Const | CPF_Native | CPF_Transient)
TArray<class UClass*>                          PendingInjectionClasses;                      // 0x0138 (0x0010)
[0x0000000000402002] (CPF_Const | CPF_Transient | CPF_NeedCtorLink)
TArray<struct FObjectProviderPendingCallback>  PendingCallbacks;                             // 0x0148
(0x0010) [0x0000000000402002] (CPF_Const | CPF_Transient | CPF_NeedCtorLink)
uint8_t                                        bTriggeringCallbacks;                         // 0x0158 (0x0001)
[0x0000000000002002] (CPF_Const | CPF_Transient)
class UObjectProvider*                         Parent;                                       // 0x0160 (0x0008)
[0x000000000408200A] (CPF_Const | CPF_ExportObject | CPF_Transient | CPF_Component |
CPF_EditInline)
TArray<class UObjectProvider*>                 Children;                                     // 0x0168 (0x0010)
[0x000000000448200A] (CPF_Const | CPF_ExportObject | CPF_Transient | CPF_Component |
CPF_NeedCtorLink | CPF_EditInline)
TArray<class UObjectProvider*>                 Proxies;                                      // 0x0178 (0x0010)
[0x000000000448200A] (CPF_Const | CPF_ExportObject | CPF_Transient | CPF_Component |
CPF_NeedCtorLink | CPF_EditInline)
TArray<struct FObjectProviderPendingCallback>  PendingInjectionCallbacks;                    //
0x0188 (0x0010) [0x0000000000402002] (CPF_Const | CPF_Transient | CPF_NeedCtorLink)
struct FScriptDelegate                         __ObjectSubscriptionCallback__Delegate;       // 0x0198
(0x0018) [0x0000000000400000] (CPF_NeedCtorLink)
struct FScriptDelegate                         __ObjectListSubscriptionCallback__Delegate;   // 0x01B0
(0x0018) [0x0000000000400000] (CPF_NeedCtorLink)
struct FScriptDelegate                         __ObjectChangeCallback__Delegate;             // 0x01C8
(0x0018) [0x0000000000400000] (CPF_NeedCtorLink)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.ObjectProvider");
}
```

```
return uClassPointer;
};

void SetParent(class UObjectProvider* InParent);
void RemoveProxy(class UObjectProvider* InProxy);
void AddProxy(class UObjectProvider* InProxy);
void SetSingleton(class UClass* ObjClass, class UObject* Replacement);
void Replace(class UObject* Existing, class UObject* Replacement);
void AddAndRemoveObjects(TArray<class UObject*>& AddObjects, TArray<class UObject*>&
RemoveObjects);
void RemoveObjects(TArray<class UObject*>& InObjects);
void RemoveAllObjects(class UClass* ObjectClass);
void RemoveObject(class UObject* Obj);
void AddObjects(TArray<class UObject*>& InObjects);
void AddObject(class UObject* Obj);
int32_t GetExactCount(class UClass* ObjClass);
int32_t GetCount(class UClass* ObjClass);
class UObject* GetOrCreate(class UClass* ObjClass);
class UObject* GetExact(class UClass* ObjClass);
class UObject* GetUnsafe(class UClass* ObjClass);
class UObject* Get(class UClass* ObjClass);
void AllObjects(class UClass* BaseClass, class UClass* InterfaceClass, class UObject*& Obj);
bool IsRegisteredForInjection(class UObject* Subscriber);
void InjectDelayed(class UObject* Subscriber);
void Inject(class UObject* Subscriber);
void UnsubscribeAll(class UObject* Subscriber);
void Unsubscribe(struct FScriptDelegate Callback, struct FScriptDelegate Callback2);
void SubscribeList(class UClass* BaseClass, struct FScriptDelegate Callback);
void SubscribeOnce(class UClass* BaseClass, struct FScriptDelegate OnAdd, struct
FScriptDelegate OnRemove);
void Subscribe(class UClass* BaseClass, struct FScriptDelegate OnAdd, struct FScriptDelegate
OnRemove);
void ObjectChangeCallback();
void ObjectListSubscriptionCallback(class UObjectProvider* Provider);
void ObjectSubscriptionCallback(class UObject* Obj);
};

// Class Core.DistributionVector
// 0x000C (0x0070 - 0x007C)
class UDistributionVector : public UComponent
{
public:
struct FPointer                                VfTable_FCurveEdInterface;                 // 0x0070 (0x0008)
[0x0000000000801002] (CPF_Const | CPF_Native | CPF_NoExport)
unsigned long                                  bCanBeBaked : 1;                           // 0x0078 (0x0004)
[0x0000000000000001] [0x00000001] (CPF_Edit)
unsigned long                                  bIsDirty : 1;                              // 0x0078 (0x0004)
[0x0000000000000000] [0x00000002]

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;
```

```
if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.DistributionVector");
}

return uClassPointer;
};

struct FVector GetVectorValue(float F, int32_t LastExtreme);
};

// Class Core.DistributionFloat
// 0x000C (0x0070 - 0x007C)
class UDistributionFloat : public UComponent
{
public:
struct FPointer                                VfTable_FCurveEdInterface;                 // 0x0070 (0x0008) [0x0000000000801002] (CPF_Const | CPF_Native | CPF_NoExport)
unsigned long                                  bCanBeBaked : 1;                           // 0x0078 (0x0004) [0x0000000000000001] [0x00000001] (CPF_Edit)
unsigned long                                  bIsDirty : 1;                              // 0x0078 (0x0004) [0x0000000000000000] [0x00000002]

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.DistributionFloat");
}

return uClassPointer;
};

float GetFloatValue(float F);
};

// Class Core.Commandlet
// 0x0054 (0x0060 - 0x00B4)
class UCommandlet : public UObject
{
public:
class FString                                  HelpDescription;                           // 0x0060 (0x0010) [0x0000000000408002] (CPF_Const | CPF_Localized | CPF_NeedCtorLink)
class FString                                  HelpUsage;                                 // 0x0070 (0x0010) [0x0000000000408002] (CPF_Const | CPF_Localized | CPF_NeedCtorLink)
class FString                                  HelpWebLink;                               // 0x0080 (0x0010) [0x0000000000408002] (CPF_Const | CPF_Localized | CPF_NeedCtorLink)
TArray<class FString>                          HelpParamNames;                            // 0x0090 (0x0010) [0x0000000000408002] (CPF_Const | CPF_Localized | CPF_NeedCtorLink)
TArray<class FString>                          HelpParamDescriptions;                     // 0x00A0 (0x0010)
```

[0x0000000000408002] (CPF_Const | CPF_Localized | CPF_NeedCtorLink)
unsigned long IsServer : 1; // 0x00B0 (0x0004) [0x0000000000000000] [0x00000001]
unsigned long IsClient : 1; // 0x00B0 (0x0004) [0x0000000000000000] [0x00000002]
unsigned long IsEditor : 1; // 0x00B0 (0x0004) [0x0000000000000000] [0x00000004]
unsigned long LogToConsole : 1; // 0x00B0 (0x0004) [0x0000000000000000] [0x00000008]
unsigned long ShowErrorCount : 1; // 0x00B0 (0x0004) [0x0000000000000000] [0x00000010]

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Commandlet");
}

return uClassPointer;
};

int32_t eventMain(class FString Params);
};

// Class Core.HelpCommandlet
// 0x0004 (0x00B4 - 0x00B8)
class UHelpCommandlet : public UCommandlet
{
public:

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.HelpCommandlet");
}

return uClassPointer;
};

int32_t eventMain(class FString Params);
};

// Class Core.Breadcrumbs
// 0x0008 (0x0060 - 0x0068)
class UBreadcrumbs : public UObject
{

```
public:
struct FPointer                                BreadcrumbInstance;                    // 0x0060 (0x0008)
[0x0000000000001000] (CPF_Native)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Breadcrumbs");
}

return uClassPointer;
};

void BreadcrumbFloat(class FString Category, float Value);
void BreadcrumbString(class FString Category, class FString Value);
};

// Class Core.Base64
// 0x0000 (0x0060 - 0x0060)
class UBase64 : public UObject
{
public:

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Base64");
}

return uClassPointer;
};

static void DecodeStringInline(class FString Input, TArray<uint8_t>& Output);
static TArray<uint8_t> DecodeString(class FString Input);
static void DecodeInline(TArray<uint8_t>& Input, TArray<uint8_t>& Output);
static TArray<uint8_t> Decode(TArray<uint8_t>& Input);
static void EncodeStringInline(TArray<uint8_t>& Input, class FString& Output);
static class FString EncodeString(TArray<uint8_t>& Input);
static void EncodeInline(TArray<uint8_t>& Input, TArray<uint8_t>& Output);
static TArray<uint8_t> Encode(TArray<uint8_t>& Input);
};

// Class Core.AutomationTest
// 0x0040 (0x0060 - 0x00A0)
class UAutomationTest : public UObject
{
```

```
public:
TArray<class FString>                    MaterialsCompiled;                    // 0x0060 (0x0010) [0x0000000000400000] (CPF_NeedCtorLink)
TArray<class FString>                    MaterialsFailedCompile;               // 0x0070 (0x0010) [0x0000000000400000] (CPF_NeedCtorLink)
TArray<class FString>                    AsyncPreloadPackagesMissing;          // 0x0080 (0x0010) [0x0000000000400000] (CPF_NeedCtorLink)
TArray<struct FScriptWarning>            ScriptWarnings;                       // 0x0090 (0x0010) [0x0000000000400000] (CPF_NeedCtorLink)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.AutomationTest");
}

return uClassPointer;
};

};

// Class Core.AsyncTask
// 0x0070 (0x0060 - 0x00D0)
class UAsyncTask : public UObject
{
public:
unsigned long                            bComplete : 1;                        // 0x0060 (0x0004) [0x0000004000000000] [0x00000001]
unsigned long                            bDisposed : 1;                        // 0x0060 (0x0004) [0x0000004000000000] [0x00000002]
class UError*                            Error;                                // 0x0068 (0x0008) [0x0000004000000000]
struct FScriptDelegate                   __EventAsyncTaskSuccess__Delegate;    // 0x0070 (0x0018) [0x0000000000400000] (CPF_NeedCtorLink)
struct FScriptDelegate                   __EventAsyncTaskFail__Delegate;       // 0x0088 (0x0018) [0x0000000000400000] (CPF_NeedCtorLink)
struct FScriptDelegate                   __EventAsyncTaskComplete__Delegate;   // 0x00A0 (0x0018) [0x0000000000400000] (CPF_NeedCtorLink)
struct FScriptDelegate                   __EventDisposed__Delegate;            // 0x00B8 (0x0018) [0x0000000000400000] (CPF_NeedCtorLink)

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.AsyncTask");
}
```

```
return uClassPointer;
};

void QueCallbacks();
static class UAsyncTask* CreateError(class UError* InError);
static class UAsyncTask* CreateSuccess();
static class UAsyncTask* Create();
class UAsyncTask* Watch(class UAsyncTask* Other);
static class UAsyncTask* All(TArray<class UAsyncTask*> Dependents);
class UAsyncTask* DependOn(class UAsyncTask* Other);
class UAsyncTask* eventNotifyOnDispose(struct FScriptDelegate Callback);
void eventClearCallbacks();
void eventDispose();
void SetComplete(class UError* InError);
void eventSetError(class UError* InError);
class UAsyncTask* eventNotifyOnComplete(struct FScriptDelegate Callback);
class UAsyncTask* eventNotifyOnFail(struct FScriptDelegate Callback);
class UAsyncTask* eventNotifyOnSuccess(struct FScriptDelegate Callback);
void EventDisposed();
void EventAsyncTaskComplete(class UError* TaskError);
void EventAsyncTaskFail(class UError* TaskError);
void EventAsyncTaskSuccess();
};

// Class Core.AsyncResult
// 0x0000 (0x00D0 - 0x00D0)
class UAsyncResult : public UAsyncTask
{
public:

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.AsyncResult");
}

return uClassPointer;
};

};

// Class Core._Types_Core
// 0x0000 (0x0060 - 0x0060)
class U_Types_Core : public UObject
{
public:

public:
static UClass* StaticClass()
```

```
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core._Types_Core");
}

return uClassPointer;
};

};

// Class Core.State
// 0x0060 (0x0130 - 0x0190)
class UState : public UStruct
{
public:
uint8_t                                    UnknownData00[0x60];                      // 0x0130 (0x0060)
MISSED OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.State");
}

return uClassPointer;
};

};

// Class Core.Package
// 0x00E8 (0x0060 - 0x0148)
class UPackage : public UObject
{
public:
uint8_t                                    UnknownData00[0xE8];                      // 0x0060 (0x00E8)
MISSED OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Package");
}
```

```
return uClassPointer;
};

};

// Class Core.Class
// 0x0228 (0x0190 - 0x03B8)
class UClass : public UState
{
public:
uint8_t                                    UnknownData00[0x228];                     // 0x0190 (0x0228)
MISSED OFFSET

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.Class");
}

return uClassPointer;
};

};

// Class Core.__AsyncTask__All_0x1
// 0x0010 (0x0060 - 0x0070)
class U__AsyncTask__All_0x1 : public UObject
{
public:
int32_t                                    DependentsCount;                          // 0x0060 (0x0004)
[0x0000000000000000]
class UAsyncTask*                          Parent;                                   // 0x0068 (0x0008)
[0x0000000000000000]

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.__AsyncTask__All_0x1");
}

return uClassPointer;
};

void __AsyncTask__All_0x1();
};
```

```
// Class Core._LoggingDoc
// 0x0000 (0x0060 - 0x0060)
class U_LoggingDoc : public UObject
{
public:

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core._LoggingDoc");
}

return uClassPointer;
};

static void TestSpecialLogging();
};

// Class Core._Types_Generated
// 0x0000 (0x0060 - 0x0060)
class U_Types_Generated : public UObject
{
public:

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core._Types_Generated");
}

return uClassPointer;
};

};

// Class Core.ArrayFuncs
// 0x0000 (0x0060 - 0x0060)
class UArrayFuncs : public UObject
{
public:

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;
```

```
if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.ArrayFuncs");
}

return uClassPointer;
};

static void GetRandomElement();
static void ShuffleArray();
};

// Class Core.IDisposable
// 0x0000 (0x0060 - 0x0060)
class UIDisposable : public UInterface
{
public:

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.IDisposable");
}

return uClassPointer;
};

void eventDispose();
};

// Class Core.RotatorConversions
// 0x0000 (0x0060 - 0x0060)
class URotatorConversions : public UObject
{
public:

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.RotatorConversions");
}

return uClassPointer;
};

static struct FRotatorRadians GetAsRadians(struct FRotator InRotator);
```

```
static struct FRotatorDegrees GetAsDegrees(struct FRotator InRotator);
};

// Class Core.TAsyncResult
// 0x0000 (0x00D0 - 0x00D0)
class UTAsyncResult : public UAsyncTask
{
public:

public:
static UClass* StaticClass()
{
static UClass* uClassPointer = nullptr;

if (!uClassPointer)
{
uClassPointer = UObject::FindClass("Class Core.TAsyncResult");
}

return uClassPointer;
};

};

/*
#
================================================================================
=========== #
#
#
================================================================================
=========== #
*/

#ifdef _MSC_VER
#pragma pack(pop)
#endif
```

Removed: 891
Added: 81