VOYAGER

User Guide

VERSION 8.0

ROSS

Thank You for Choosing Ross

You've made a great choice. We expect you will be very happy with your purchase of Ross Technology.

Our mission is to:

1. Provide a Superior Customer Experience
    - offer the best product quality and support
2. Make Cool Practical Technology
    - develop great products that customers love

Ross has become well known for the Ross Video Code of Ethics. It guides our interactions and empowers our employees. I hope you enjoy reading it below.

If anything at all with your Ross experience does not live up to your expectations be sure to reach out to us at solutions@rossvideo.com.

David Ross

CEO, Ross Video

dross@rossvideo.com

Ross Video Code of Ethics

Any company is the sum total of the people that make things happen. At Ross, our employees are a special group. Our employees truly care about doing a great job and delivering a high quality customer experience every day. This code of ethics hangs on the wall of all Ross Video locations to guide our behavior:

1. We will always act in our customers' best interest.
2. We will do our best to understand our customers' requirements.
3. We will not ship crap.
4. We will be great to work with.
5. We will do something extra for our customers, as an apology, when something big goes wrong and it's our fault.
6. We will keep our promises.
7. We will treat the competition with respect.
8. We will cooperate with and help other friendly companies.
9. We will go above and beyond in times of crisis. *If there's no one to authorize the required action in times of company or customer crisis - do what you know in your heart is right. (You may rent helicopters if necessary.)*

Voyager User Guide

- Ross Part Number: 3808DR-001-8.0 Rev 1
- Version: 8.0
- Date/Time: 3/12/2026 5:04 PM

The information contained in this guide is subject to change without notice or obligation.

Copyright

Patents

Patent numbers US 7,034,886; US 7,508,455; US 7,602,446; US 7,802,802 B2; US 7,834,886; US 7,914,332; US 8,307,284; US 8,407,374 B2; US 8,499,019 B2; US 8,519,949 B2; US 8,743,292 B2; GB 2,419,119 B; GB 2,447,380 B; and other patents pending.

Notice

The material in this manual is furnished for informational use only. It is subject to change without notice and should not be construed as commitment by Ross Video Limited. Ross Video Limited assumes no responsibility or liability for errors or inaccuracies that may appear in this manual.

End User Software License Agreement

This End User Software License Agreement is a legal agreement between you (the "**Licensee**") and Ross Video Limited ("**Ross Video**") specifying the terms and conditions of your installation and use of the Software and all Documentation (as those terms are defined herein).

IMPORTANT:

BY DOWNLOADING, ACCESSING, INSTALLING OR USING THE SOFTWARE AND/OR DOCUMENTATION AND/OR BY AUTHORIZING ANY THIRD PARTY, INCLUDING WITHOUT LIMITATION AN INSTALLER OR COMMISSIONER ACTING ON YOUR BEHALF TO DO SO, LICENSEE AGREES TO THE TERMS OF THIS AGREEMENT AND THE LICENSE GRANTED HEREUNDER SHALL BE EFFECTIVE AS OF AND FROM SUCH DATE. IF YOU DO NOT WISH TO ACCEPT THE TERMS AND CONDITIONS OF THIS AGREEMENT, DO NOT DOWNLOAD, ACCESS, INSTALL, REFER TO OR OTHERWISE USE THE SOFTWARE AND/OR DOCUMENTATION.

1. **INTERPRETATION.** In this Agreement, (a) words signifying the singular number include the plural and vice versa, and words signifying gender include all genders; (b) every use of the words "herein", "hereof", "hereto" "hereunder" and similar words shall be construed to refer to this Agreement in its entirety and not to any particular provision hereof; (c) reference to any agreement or other document herein will be construed as referring to such agreement or other document as from time to time amended, modified or supplemented (subject to any restrictions on such amendment, modification or supplement set forth therein); (d) every use of the words "including" or "includes" is to be construed as meaning "including, without limitation" or "includes, without limitation", respectively; and (e) references to an Article or a Section are to be construed as references to an Article or Section of or to this Agreement unless otherwise specified.

2. **DEFINITIONS.** In this Agreement, in addition to the terms defined elsewhere in this Agreement, the following terms have the meanings set out below:

   "**Affiliate**" means, with respect to any Person, any other Person who directly or indirectly controls, is controlled by, or is under direct or indirect common control with, such Person. A Person shall be deemed to control a Person if such Person possesses, directly or indirectly, the power to direct or cause the direction of the management and policies of such Person, whether through the ownership of voting securities, by contract or otherwise; and the term "controlled" and "controlling" shall have a similar meaning.

   "**Agreement**" means this End User Software License Agreement including the recitals hereto, as the same may be amended from time to time in accordance with the provisions hereof.

   "**Backup System**" means the secondary piece of Designated Equipment upon which the Software is installed and mirrored for the sole purpose of replacing a Primary System in the event such Primary System is not available or functioning properly for any reason.

   "**Change of Control**" means (a) the direct or indirect sale, transfer or exchange by the shareholders of a Party of more than fifty percent (50%) of the voting securities of such Party, (b) a merger or amalgamation or reorganization or other transaction to which a Party is party after which the shareholders of such Party immediately prior to such transaction hold less than fifty percent (50%) of the voting securities of the surviving entity, (c) the sale, exchange, or transfer of all or substantially all of the assets of a Party.

"Confidential Information" means all data and information relating to the business and management of either Party, including the Software, trade secrets and other technology to which access is obtained or granted hereunder by the other Party, and any materials provided by Ross Video to Licensee; provided, however, that Confidential Information shall not include any data or information which:

(i) is or becomes publicly available through no fault of the other Party;

(ii) is already in the rightful possession of the other Party prior to its receipt from the other Party;

(iii) is already known to the receiving Party at the time of its disclosure to the receiving Party by the disclosing Party and is not the subject of an obligation of confidence of any kind;

(iv) is independently developed by the other Party;

(v) is rightfully obtained by the other Party from a third party; or

(vi) is disclosed with the written consent of the Party whose information it is.

"Designated Equipment" shall mean (a) the hardware products sold by Ross Video to Licensee on which the Software is installed and licensed for use, as the same may be replaced from time to time by Ross Video; or (b) in the case of Software licensed on a stand-alone basis, the equipment of Licensee on which the Software is to be installed and meets the minimum specifications set out in the Documentation.

"Documentation" shall mean manuals, instruction guides, user documentation and other related materials of any kind pertaining to the Software (whether in electronic, hard-copy or other media format) that are furnished to Licensee by or on behalf of Ross Video in relation to the Software.

"Freeware" means Software that is available free of charge from Ross Video, and includes, without limitation the master control system software known as "DashBoard".

"Governmental Authority" means (a) any federal, provincial, state, local, municipal, regional, territorial, aboriginal, or other government, governmental or public department, branch, ministry, or court, domestic or foreign, including any district, agency, commission, board, arbitration panel or authority and any subdivision of any of them exercising or entitled to exercise any administrative, executive, judicial, ministerial, prerogative, legislative, regulatory, or taxing authority or power of any nature; and (b) any quasi-governmental or private body exercising any regulatory, expropriation or taxing authority under or for the account of any of them, and any subdivision of any of them.

"Improvements" means all inventions, works, discoveries, improvements and innovations of or in connection with the Software, including error corrections, bug fixes, patches and other updates in Object Code form to the extent made available to Licensee in accordance with Ross Video's release schedule.

"License Fee" means the fee(s), if any, payable in respect of the Software in accordance with the relevant invoice(s) or other purchase documents delivered in connection with this Agreement.

"License Period" means the period of time that Licensee will have the rights granted under this Agreement, as may be specified in an Order.

"**Modifications**" means any enhancements, changes, corrections, translations, adaptations, revisions, developments, upgrades or updates thereto; and "Modify" shall mean the creation of any of the foregoing.

"**Object Code**" means the machine readable executable form of a computer software program.

"Open Source Components" means third party Open Source software, libraries or other components.

"Open Source License" means the license that governs each Open Source Component.

"Order" means the documents provided by Ross Video to Licensee detailing the Ross Video products contemplated for purchase, the corresponding fees and License Period that may apply to the Software, including any and all quotations, purchase orders, acknowledgments, pro formas, invoices and other purchase documentation.

"**Parties**" means both Ross Video and Licensee and "Party" means either one of them as the context requires.

"**Person**" will be broadly interpreted and includes (a) a natural person, whether acting in his or her own capacity, or in his or her capacity as executor, administrator, estate trustee, trustee or personal or legal representative; (b) a corporation or a company of any kind, a partnership of any kind, a sole proprietorship, a trust, a joint venture, an association, an unincorporated association, an unincorporated syndicate, an unincorporated organization or any other association, organization or entity of any kind; and (c) a Governmental Authority.

"**Primary System**" means the Designated Equipment upon which the Software is installed and executed to deliver its intended functionality.

"Released Claims" has the meaning ascribed to it in Section 9(b).

"Released Parties" has the meaning ascribed to it in Section 9(b).

"Ross Video" means Ross Video Limited and its Affiliates.

"Software" means the version of the Object Code licensed and delivered to Licensee by Ross Video concurrently with delivery of this Agreement, including without limitation the Freeware, and any subsequent error corrections, updates, Modifications or Improvements provided to Licensee by Ross Video pursuant to this Agreement, but specifically excluding any features or plug-ins that may be purchased by you directly from third parties as upgrades or enhancements to the Software.

"**Source Code**" means the human readable form of a computer software program, and all tools and documentation necessary for a reasonably competent computer programmer to understand, maintain and Modify the Software.

"**Third Party Software**" means those portions of the Software, if any, which are owned or controlled by third parties and licensed to Ross Video pursuant to certain license agreements or arrangements with such third parties, including the Open Source Components and NDI®[1] software (http://NDI.video); and

"**Use**" means to execute, run, display, store, merge, network, Modify, translate, host or integrate with Licensee's products or other third party software.

[1] NDI® is a registered trademark of Vizrt NDI AB.

3. **LICENSE.** Subject to the terms and conditions of this Agreement, upon payment of the applicable License Fee by Licensee, or, in the case of Freeware only, upon download of the Software by Licensee onto its Designated Equipment, Ross Video hereby grants to Licensee a non-transferable and non-exclusive right to Use the Software and Documentation solely for the internal use of Licensee (the "License"), during the License Period. In the event that a License Period is not identified on the Order, such License Period shall be deemed to be perpetual, subject to paragraph 10 (c) of this Agreement. The Software shall only be used in connection with or installed on the Designated Equipment and, where applicable, shall only be used on the Primary System, provided such Primary System is operating properly."

   If the Primary System is not operating properly for any reason, the Software may be used on the designated Backup System for that Primary System until such time that the Primary System begins operating properly. The Software and Documentation are provided to Licensee for the exclusive use by Licensee's organization for its ordinary business purposes and shall not be used by any third party for any purposes. Licensee may make copies of the Software as required for internal backup and archival purposes. To the extent permitted hereunder, Licensee may distribute copies of the Software and/or Documentation to members of its organization, provided (a) this Agreement is included with each copy, (b) any member of its organization who uses the Software and/or Documentation accepts and agrees to be bound by the terms of this Agreement and by any other license agreements or other agreement incorporated by reference into this Agreement, and (c) Licensee has paid any applicable additional License Fees in respect of copying and redistributing of the Software. To the extent Licensee is permitted to make copies of the Software under this Agreement, Licensee agrees to reproduce and include on any copy made or portion merged into another work, all Ross Video proprietary notices, including any notices with respect to copyrights, trademarks and this License. With the exception of copying the Software for backup or archival purposes, Licensee agrees to keep a record of the number and location of all such copies and will make such record available at Ross Video's request. The Software may include mechanisms to limit or inhibit copying.

4. **LICENSE RESTRICTIONS.** Except as otherwise provided in section 3 above, Licensee shall not: (1) copy any Software or Documentation, or part thereof, which is provided to Licensee by Ross Video pursuant to this Agreement, in Object Code form, Source Code form or other human or machine readable form, including written or printed documents, without the prior written consent of Ross Video; (2) in any way market, distribute, export, translate, transmit, merge, Modify, transfer, adapt, loan, rent, lease, assign, share, sub-license, sell, make available for download on any website or make available to another Person, the Software and/or Documentation, in whole or in part, provided that Licensee shall not be prohibited from renting or leasing the Software if Ross Video has consented, in writing, to Licensee engaging in such activities in respect of the Software; (3) reverse engineer, decompile or disassemble the Software or electronically transfer it into another computer language; or (4) use the Software or Documentation in a manner that is inconsistent with the License granted hereunder or that will result in a breach of this Agreement. Licensee agrees to take all reasonable precautions to prevent third parties from using the Software and/or Documentation in any way that would constitute a breach of this Agreement, including such precautions Licensee would ordinarily take to protect its own proprietary software, hardware or information.

5. **DELIVERY.** Ross Video shall deliver to Licensee one (1) master copy of the Software in compiled binary (executable) form suitable for reproduction in electronic files only and Ross Video shall deliver to Licensee a minimum of one copy of the Documentation.

6. **IMPROVEMENTS.** Licensee may from time to time request Ross Video to incorporate certain Improvements into the Software. Ross Video may, in its sole discretion, undertake to incorporate and provide such Improvements to Licensee with or without payment of a fee to be negotiated at the time of such request. All Improvements, whether recommended and developed by Ross Video or Licensee, shall be considered the sole property of Ross Video and shall be used by Licensee pursuant to the terms of the License granted under this Agreement.

7. **FREEWARE.** Other than the obligation to pay a License Fee, which does not apply to the Freeware, all other provisions of this End User Software License Agreement apply to the Freeware in the same way as they apply to all other Software that is the subject of this Agreement. In addition, in connection with the Freeware, the following provisions apply:

   a. Licensee will not Use the Freeware to engage in or allow others to engage in any illegal activity.

   b. Licensee will not Use the Freeware in any way that will interfere with or damage the operation of the services of any third parties by overburdening/disabling network resources through automated queries, excessive usage or similar conduct.

   c. Licensee will not Use the Freeware to engage in any activity that will violate the rights of others, including, without limitation, by using it for operations that involve child labour, suppressing the right of freedom of expression or endangering the security of person.

8. **OWNERSHIP.** The Parties acknowledge and agree that, as between the Parties, Ross Video shall be the owner of all intellectual property rights in the Software, Documentation and all related Modifications and Improvements, written materials, logos, trademarks, trade names, copyright, patents, trade secrets and moral rights, registered or unregistered. No proprietary interest or title in or to the intellectual property in the Software, Documentation or any Improvements or Modifications is transferred to Licensee by this Agreement. Ross Video reserves all rights not expressly licensed to Licensee under section 3.

9. **OPEN SOURCE SOFTWARE.**

   a. Software may use and/or be provided with Open Source Components, including those detailed in the Third Party section below. To the extent stipulated by its Open Source License, each such Open Source Component is licensed directly to Licensee from its respective licensors and not sub-licensed to Licensee by Ross Video, and such Open Source Component is subject to its respective Open Source License, and not to this Agreement. If, and to the extent, an Open Source Component requires that this Agreement effectively impose, or incorporate by reference, certain disclaimers, permissions, provisions, prohibitions or restrictions, then such disclaimers, permissions, provisions, prohibitions or restrictions shall be deemed to be imposed, or incorporated by reference into this Agreement, as required, and shall supersede any conflicting provision of this Agreement, solely with respect to the corresponding Open Source Component which is governed by such Open Source License.

   b. If Licensee, or another party on Licensee's behalf, modifies, replaces or substitutes any Open Source Component used in or provided with this Software, Licensee hereby fully, forever, irrevocably and unconditionally releases and discharges Ross Video, its Affiliates and its and their employees, officers, directors, resellers, distributors and representatives (collectively, "Released Parties") from any and all claims, charges, complaints, demands, actions, causes of action, suits, rights, debts, covenants, liabilities, warranties, performance and maintenance and support obligations (collectively, "Released Claims"), of every kind and nature, with respect to such Software, including without limitation any such Released Claims that arise as a matter of applicable Law.

   c. If an Open Source License requires that the source code of its corresponding Open Source Component be made available to Licensee, and such source code was not delivered to Licensee with the Software, then Ross Video hereby extends a written offer, valid for the period prescribed in such Open Source License, to obtain a copy of the source code of the corresponding Open Source Component, from Ross Video from https://www.rossvideo.com/open-source-information/.

10. **THIRD PARTY SOFTWARE.**

a. Licensee acknowledges that the Third Party Software is not owned by Ross Video. Notwithstanding any other provision of this Agreement, Ross Video, to the extent permitted by applicable law, offers no warranties (whether express, implied, statutory or by course of communication or dealing with Licensee, or otherwise) with respect to the Third Party Software. Ross Video may pass through to Licensee, if and to the extent permitted by applicable law, any warranties expressly provided by such third parties to Ross Video for such Third Party Software.

b. FFmpeg Notice. The Software may utilize FFmpeg video components and their included libraries. FFmpeg is a trademark of Fabrice Bellard (originator of the FFmpeg project). Ross Video disclaims any ownership claim to FFmpeg. Please refer to http://ffmpeg.org (copyright is in the FFmpeg developers). FFmpeg is licensed under the GNU Lesser General Public License v2.1 or Lesser General Public License v3.0. GNU Lesser General Public License contact information: Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA. Licensee can distribute it and/or modify it under the terms of such licenses.

11. **INTELLECTUAL PROPERTY INDEMNITY.**

a. Ross Video agrees to defend, indemnify and hold harmless Licensee from final damages awarded by a court of competent jurisdiction (hereinafter referred to as the "**Losses**"), which Licensee, or any of its officers or directors, may incur, suffer or become liable for as a result of, or in connection with, any third party claim asserted against Licensee to the extent such claim is based on a contention that the Software, Documentation or any portion thereof, infringes any valid, registered, enforceable patents, copyrights, trade secrets, trademarks or other intellectual property rights of any third party, provided that (a) the allegedly infringing Software or Documentation has been used within the scope of and in accordance with the terms of this Agreement, and (b) Licensee notifies Ross Video in writing of such claim within ten (10) days of a responsible officer of Licensee becoming aware of such claim. If the Software, Documentation or any portion thereof is held to constitute an infringement of a third party's intellectual property rights, and use thereof is enjoined, Ross Video shall, at its election and expense, either (i) procure the right to use the infringing element of the Software or Documentation; or (ii) replace or modify the element of the Software or Documentation so that the infringing portion is no longer infringing and still performs the same function without any material loss of functionality. Ross Video shall make every reasonable effort to correct the situation with minimal effect upon the operations of Licensee.

b. Notwithstanding the above, Ross Video reserves the right to terminate this Agreement and the License granted hereunder on immediate notice to Licensee, and without liability to Licensee, in the event that the Software or Documentation constitutes or may, in Ross Video's determination, constitute, an infringement of the rights of a third party that Ross Video, in its sole discretion, does not consider to be affordably remediable.

c. Either party may terminate this Agreement immediately should any Software become, or in either party's opinion be likely to become, the subject of a claim of infringement of any intellectual property right and, in such event, there shall be no claim by either Licensee or Ross Video against the other arising out of such termination, provided that the foregoing shall not apply to a claim for infringement by Ross Video against Licensee in the event that Licensee is alleged to have infringed Ross Video's intellectual property rights, in which case Licensee shall remain liable for all outstanding License Fees and other amounts owing to Ross Video.

d. Notwithstanding the foregoing, Ross Video shall have no liability for any claim of infringement based on use of other than a current, unaltered release of the Software and/or Documentation available from Ross Video if such infringement would have been avoided by the use of a current, unaltered release of the Software and/or Documentation provided that such current, unaltered release performs substantially in conformity with the specifications set out in the Documentation and was provided, at no additional cost by Ross Video, to those subscribing for maintenance services for the Software or Documentation.

12. **CONFIDENTIALITY.** Each Party shall maintain in confidence all Confidential Information of the other Party, shall use such Confidential Information only for the purpose of exercising its rights and fulfilling its obligations under this Agreement, and shall not disclose any Confidential Information of the disclosing Party to any third party except as expressly permitted hereunder or make any unauthorized use thereof. Each Party shall disclose the Confidential Information only to those of its employees, consultants, advisors, and/or subcontractors who have a need to know the Confidential Information. Each Party shall, prior to disclosing the Confidential Information to such employees, consultants, advisors and/or subcontractors, obtain their agreement to receive and use the Confidential Information on a confidential basis on the same terms and conditions contained in this Agreement. The receiving Party shall treat the Confidential Information of the disclosing Party with the same degree of care against disclosure and/or unauthorized use as it affords to its own information of a similar nature, or a reasonable degree of care, whichever is greater. The receiving Party further agrees not to remove or destroy any proprietary or confidential legends or markings placed upon any documents or other materials of the disclosing Party. The obligations of confidence set forth in this Agreement shall extend to any Affiliates that have received Confidential Information of the disclosing Party and shall also cover Confidential Information disclosed by any Affiliate. The receiving Party shall be responsible for any actions or omissions of its Affiliates as if such actions or omissions were its own.

    Either party may disclose certain Confidential Information if it is expressly required to do so pursuant to legal, judicial, or administrative proceedings, or otherwise required by law, provided that (i) such Party provides the other Party with reasonable written notice prior to such disclosure; (ii) such Party seeks confidential treatment for such Confidential Information; (iii) the extent of such disclosure is only to the extent expressly required by law or under the applicable court order; and (iv) such Party complies with any applicable protective or equivalent order.

    Each of Ross Video and Licensee (the **"Indemnifying Party"**, as applicable) agree to indemnify the other (the **"Indemnified Party"**, as applicable) for all Losses incurred by the Indemnified Party as a result of a failure of the Indemnifying Party to comply with its obligations under this Section 12 provided that the Indemnified Party has given prompt notice of any such claim and, to the extent that a claim may lie against a third party for the unauthorized disclosure of such Confidential Information, the right to control and direct the investigation, preparation, action and settlement of each such claim and, further, provided that the Indemnified Party reasonably co-operates with the Indemnifying Party in connection with the foregoing and provides the Indemnifying Party with all information in the Indemnified Party's possession related to such claim and such further assistance as reasonably requested by the Indemnifying Party.

    The Parties acknowledge and agree that any breach of the confidentiality provisions of this Agreement by one Party may cause significant and irreparable injury to the other Party that is not compensable monetarily, as well as damages that may be difficult to ascertain, and agrees that, in addition to such other remedies that may be available at law or in equity, the other Party shall be entitled to seek injunctive relief (including temporary restraining orders, interim injunctions and permanent injunctions) in a court of competent jurisdiction in the event of the breach or threatened breach by such party of any of the confidentiality provisions of this Agreement. The relief contemplated in this Section shall be available to each Party without the necessity of having to prove actual damages and without the necessity of having to post any bond or other security. Each Party further agrees to notify the other Party in the event that it learns of or has reason to believe that any Person has breached the confidentiality provisions of this Agreement.

13. **LIMITATION OF LIABILITY.** The limitation of liability provisions of this Agreement reflect an informed voluntary allocation of the risks (known and unknown) that may exist in connection with the licensing of the Software or Documentation hereunder by Ross Video, and that voluntary risk allocation represents a material part of the Agreement reached between Ross Video and Licensee. Should Ross Video be in breach of any obligation, Licensee agrees that Licensee's remedies will be limited to those set forth in this Agreement. No action, regardless of form, arising out of this Agreement may be brought by Licensee more than twelve (12) months after the facts giving rise to the cause of action have occurred, regardless of whether those facts by that time are known to, or reasonably ought to have been discovered by, Licensee.

    (A) EXCEPT AS EXPRESSLY PROVIDED IN THIS AGREEMENT, THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" AND ROSS VIDEO (I) MAKES NO OTHER REPRESENTATIONS, AND PROVIDES NO WARRANTIES OR CONDITIONS OF ANY KIND, EXPRESS OR IMPLIED, STATUTORY, BY USAGE OF TRADE CUSTOM OF DEALING, OR OTHERWISE, AND (II) SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES INCLUDING ANY IMPLIED WARRANTY OF UNINTERRUPTED OR ERROR FREE OPERATION, MERCHANTABILITY, QUALITY OR FITNESS FOR A PARTICULAR PURPOSE. ROSS VIDEO DOES NOT REPRESENT OR WARRANT THAT THE SOFTWARE WILL MEET ANY OR ALL OF LICENSEE'S PARTICULAR REQUIREMENTS, THAT THE USE AND OPERATION OF THE SOFTWARE WILL OPERATE ERROR-FREE OR UNINTERRUPTED, THAT ALL PROGRAMMING ERRORS IN THE SOFTWARE CAN BE FOUND IN ORDER TO BE CORRECTED, OR THAT THE SOFTWARE WILL BE COMPATIBLE WITH OTHER PROGRAMS, SYSTEMS, AND HARDWARE.

    (B) IN NO EVENT SHALL ROSS VIDEO, ITS AFFILIATES AND LICENSORS, AND THEIR RESPECTIVE DIRECTORS, OFFICERS, EMPLOYEES AND AGENTS, BE LIABLE FOR ANY CLAIM FOR INDIRECT, CONSEQUENTIAL, SPECIAL, INCIDENTAL, PUNITIVE, EXEMPLARY, AGGRAVATED DAMAGES; LOST PROFITS, OR LOST REVENUE ARISING FROM OR IN CONNECTION WITH THIS AGREEMENT, REGARDLESS OF THE FORM OF ACTION, WHETHER IN CONTRACT, OR IN TORT, EVEN IF THE PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

    (C) IN ANY EVENT THE AGGREGATE LIABILITY OF ROSS VIDEO, ITS AFFILIATES AND LICENSORS, AND THEIR RESPECTIVE DIRECTORS, OFFICERS, EMPLOYEES AND AGENTS, FOR ANY CLAIM FOR DIRECT DAMAGES WITH RESPECT TO THE SUBJECT MATTER OF THIS AGREEMENT SHALL NOT EXCEED THE AMOUNT OF THE PURCHASE PRICE PAID TO ROSS VIDEO UNDER THIS AGREEMENT.

14. **TERM AND TERMINATION**.

    (1) Unless terminated earlier in accordance with the terms of this Agreement, the term of this Agreement shall commence upon Licensee's first download, access, installation, or other use of the Software or Documentation and continues until, in the case of Software licensed with Designated Equipment provided by Ross Video, the earliest of (a) the end of the License Period, or (b) if the Designated Equipment is assigned or transferred in accordance with this Agreement, the date on which the Designated Equipment is no longer owned by Licensee;

    (2) Either Party shall have the right to terminate this Agreement on notice to the other Party if:

        (a) the other Party fails to pay any fees or other amounts when due hereunder or under any other agreement between the Parties (or any Affiliates of the Parties, as applicable) in connection with the Software and/or Designated Equipment and such breach is not cured within thirty (30) days after written notice of such failure to pay is given to the defaulting Party by the non-defaulting Party;

        (b) the other Party shall file a voluntary petition in bankruptcy or insolvency or shall petition for reorganization under any bankruptcy law, consent to an involuntary petition in bankruptcy, or if a receiving order is given against it under the Bankruptcy and Insolvency Act (Canada) or the comparable law of any other jurisdiction (and such is not dismissed within ten (10) days);

(c) there shall be entered an order, judgment or decree by a court of competent jurisdiction, upon the application of a creditor, approving a petition seeking reorganization or appointing a receiver, trustee or liquidator of all or a substantial part of the other Party's assets and such order, judgment or decree continues in effect for a period of thirty (30) consecutive days; or

(d) the other Party shall fail to perform any of the other material obligations set forth in this Agreement and such default, in the case of a default which is remediable, continues for a period of thirty (30) days after written notice of such failure has been given by the non-defaulting Party or, in the case of a non-remediable default, immediately upon notice.

(3) Notwithstanding anything to the contrary contained in this Agreement:

(a) Ross Video may forthwith terminate this Agreement if Licensee is in breach of any of sections 3, 4 or 12 of this Agreement. For greater certainty, in such instances Ross Video shall provide written notice of such termination as soon as practicable but written notice shall not be a necessary prerequisite to such termination; and

(b) in the event of a Change of Control of Licensee, Ross Video shall have the right to terminate this Agreement and the License granted hereunder upon thirty (30) days' prior written notice to Licensee. For greater certainty, Ross Video's right to terminate in the event of a Change of Control of Licensee shall continue for a period of six (6) months from the date Licensee delivers notice of such Change of Control to Ross Video.

(c) Ross Video may terminate the License immediately on the date on which it provides notice to Licensee, if its agreements for Third Party Software are terminated.

(4) Upon the termination or expiry of this Agreement:

(a) Licensee shall immediately cease and desist all use of the Software and Documentation;

(b) Licensee shall immediately deliver to Ross Video any of Ross Video's Confidential Information provided hereunder (including the Software and Documentation) then in its possession or control, if any, and shall deliver a certificate of an officer of Licensee certifying the completeness of same;

(c) Licensee shall refrain from further use of such Confidential Information; and

(d) Licensee shall forthwith pay all amounts owing to Ross Video or any of its Affiliates hereunder.

15. **SURVIVAL.** The provisions of sections 1, 2, 6, 8, 9, 10, 11, 12, 13, 14, 18, 22, 23, and 24 herein shall survive the expiry or termination of this Agreement.

16. **FORCE MAJEURE.** Dates and times by which Ross Video is required to render performance under this Agreement shall be automatically postponed to the extent and for the period that Ross Video is prevented from meeting them by reason of events of force majeure or any cause beyond its reasonable control provided Ross Video notifies Licensee of the commencement and nature of such cause and uses its reasonable efforts to render performance in a timely manner.

17. **ASSIGNMENT.** Ross Video may assign this Agreement, or any of its rights or obligations hereunder, in whole or in part, upon notice to Licensee. Licensee shall not assign this Agreement, or any of its rights or obligations hereunder, in whole or in part, without the prior written consent of Ross Video, which consent may not be unreasonably withheld. This Agreement enures to the benefit of and is binding upon each of the Parties and their respective successors and permitted assigns.

18. **GOVERNING LAW.** If Licensee acquired the Ross Product(s) in the United States or Canada, the laws of the state or province where Licensee's principal place of business is located govern the interpretation of this Agreement, claims for its breach, and all other claims regardless of conflict of laws principles. If Licensee acquired the Ross Product(s) in the European Union or the United Kingdom, then the laws of England and Wales apply. If Licensee acquired the Ross Product(s) in any other country, then the laws of the Province of Ontario, Canada shall apply.

19. **LANGUAGE.** The Parties have expressly required that this Agreement and all documents relating thereto be prepared in English. Les parties ont expressément exigé que cette convention ainsi que tous les documents qui s'y rattachent soient rédigés en anglais.

20. **GOVERNMENT CONTRACTS.** If the Software and/or Documentation to be furnished to Licensee hereunder are to be used in the performance of a government contract or subcontract, the Software and/or Documentation shall be provided on a "restricted rights" basis only and Licensee shall place a legend, in addition to applicable copyright notices, in the form provided under the applicable governmental regulations. For greater certainty, Ross Video shall not be subject to any flow-down provisions required by any customer of Licensee that is a Governmental Authority unless Ross Video expressly agrees to be bound by such flow-down provisions in writing.

21. **EXPORT AND IMPORT LAWS.** Licensee acknowledges and agrees that the Software (including any technical data and related technology) may be subject to the export control laws, rules, regulations, restrictions and national security controls of the United States and other applicable countries (the "**Export Controls**") and agrees not to export, re-export, import or allow the export, re-export or import of such export-controlled Software (including any technical data and related technology) or any copy, portion or direct product of the foregoing in violation of the Export Controls. Licensee hereby represents that it is not an entity or person to whom provision of the Software (including any technical data and related technology) is restricted or prohibited by the Export Controls. Licensee agrees that it has the sole responsibility to obtain any authorization to export, re-export, or import the Software (including any technical data and related technology), as may be required. Licensee will defend, indemnify and hold Ross Video harmless from any and all claims, losses, liabilities, damages, fines, penalties, costs and expenses (including attorney's fees) arising from or relating to any breach by Licensee of its obligations under this Section.

22. **AMENDMENT AND WAIVER.** No amendment, discharge, modification, restatement, supplement, termination or waiver of this Agreement or any Section of this Agreement is binding unless it is in writing and executed by the Party to be bound. No waiver of, failure to exercise or delay in exercising, any Section of this Agreement constitutes a waiver of any other Section (whether or not similar) nor does any waiver constitute a continuing waiver unless otherwise expressly provided.

23. **SEVERABILITY.** Each Section of this Agreement is distinct and severable. If any Section of this Agreement, in whole or in part, is or becomes illegal, invalid, void, voidable or unenforceable in any jurisdiction by any court of competent jurisdiction, the illegality, invalidity or unenforceability of that Section, in whole or in part, will not affect (a) the legality, validity or enforceability of the remaining Sections of this Agreement, in whole or in part; or (b) the legality, validity or enforceability of that Section, in whole or in part, in any other jurisdiction.

24. **ENTIRE AGREEMENT.** This Agreement, and any other documents referred to herein, constitutes the entire agreement between the Parties relating to the subject matter of this Agreement and supersedes all prior written or oral agreements, representations and other communications between the Parties.

Updated: November 1, 2023

Warranty and Repair Policy

Ross Video Limited (Ross) warrants its Voyager systems to be free from defects under normal use and service for the following time periods from the date of shipment:

- Voyager Server — 12 months
- Voyager Software Upgrades — 12 months free of charge
- System and Media hard drives — 12 months

If an item becomes defective within the warranty period Ross will repair or replace the defective item, as determined solely by Ross.

Warranty repairs will be conducted at Ross, with all shipping FOB Ross dock. If repairs are conducted at the customer site, reasonable out-of-pocket charges will apply. At the discretion of Ross, and on a temporary loan basis, plug in circuit boards or other replacement parts may be supplied free of charge while defective items undergo repair. Return packing, shipping, and special handling costs are the responsibility of the customer.

This warranty is void if products are subjected to misuse, neglect, accident, improper installation or application, or unauthorized modification.

In no event shall Ross Video Limited be liable for direct, indirect, special, incidental, or consequential damages (including loss of profit). Implied warranties, including that of merchantability and fitness for a particular purpose, are expressly limited to the duration of this warranty.

This warranty is TRANSFERABLE to subsequent owners, subject to Ross' notification of change of ownership.

Extended Warranty

For customers that require a longer warranty period, Ross offers an extended warranty plan to extend the standard warranty period by one year increments. For more information about an extended warranty for your Voyager system, contact your regional sales manager.

Environmental Information

The equipment that you purchased required the extraction and use of natural resources for its production. It may contain hazardous substances that could impact health and the environment.

To avoid the potential release of those substances into the environment and to diminish the need for the extraction of natural resources, Ross Video encourages you to use the appropriate take-back systems. These systems will reuse or recycle most of the materials from your end-of-life equipment in an environmentally friendly and health conscious manner.

The crossed-out wheeled bin symbol invites you to use these systems.

If you need more information on the collection, reuse, and recycling systems, please contact your local or regional waste administration.

You can also contact Ross Video for more information on the environmental performances of our products.

Company Address

Ross Video Limited	Ross Video Incorporated
8 John Street Iroquois, Ontario Canada, K0E 1K0	P.O. Box 880 Ogdensburg, New York USA 13669-0880

General Business Office:	(+1) 613-228-0688
Toll Free:	(+1) 844-652-0645
Fax:	(+1) 613-652-4425
Toll Free Technical Support:	1-833-859-0499 (North America) +800 3540 3545 (International)
Alternately, you can contact:	
Technical Support:	(+1) 613-686-1557
Australia/Sydney Local Support:	1300 007 677*
E-mail for Technical Support:	techsupport@rossvideo.com
E-mail for General Information:	solutions@rossvideo.com
Website:	http://www.rossvideo.com

*If the local support specialist is not available, your call will be transferred automatically to our North America center.

Contents

What's New?

The following new features were added in Voyager 8.0:

- You can now publish your Voyager projects to the XPression Project Server and then deploy them to each Voyager machine.

  ➢ Publishing and Deploying a Voyager Project

Introduction

Voyager is Ross Video's latest graphics platform, powered by Epic Games' Unreal Engine. Voyager leverages the world's most powerful and realistic renderer, enabling it's use for Augmented Reality (AR) and Virtual Studio (VS) and helping the user to create stunning, complex virtual environments designed for use in broadcast television.

Voyager uses the Lucid Studio control platform as an operator-friendly front end, so operators are not required to have in-depth knowledge of the Unreal engine to use the system. Voyager supports customization, flexibility, and scalability in terms of number of cameras and graphics engines through the integration of Lucid Studio and Lucid Track applications.

Voyager 8.0 uses Unreal Engine 5.6.0 and supports Lucid 8.0 and Voyager Trackless 8.0.

About This Guide

If, at any time, you have a question pertaining to the operation of the Voyager system, please contact Ross Video at the numbers listed in the section Getting Help. Our technical staff are always available for consultation, training, or service.

Documentation Conventions

Special text formats are used in this guide to identify parts of the user interface, text that a user must enter, or a sequence of menus and sub-menus that must be followed to reach a particular command.

Bold text	Bold text identifies a user interface element such as a dialog box, menu item, or button. For example: In the **Slug** column, type a slug name for the story.
Italic text	Italic text is used to identify the titles of referenced guides, manuals, or documents. For example: For more information, refer to the *DashBoard User Guide.*
`Courier text`	Courier text identifies text that a user must type. For example: In the **Username** box, type `postgres`.
Menu Sequences	Menu arrows are used in procedures to identify a sequence of menu items that you must follow. For example: If a step reads **Server > Save As**, you would select the **Server** menu and then select **Save As**.
Hypertext	Identifies a hyperlink to a related topic.

Getting Help

At Ross Video, we take pride in the quality of our products, but if problems occur, help is as close as the nearest telephone.

Our 24-hour Hot Line service ensures you have access to technical expertise around the clock. After-sales service and technical support is provided directly by Ross Video personnel. During business hours (Eastern Time), technical support personnel are available by telephone. After hours and on weekends, a direct emergency technical support phone line is available. If the technical support person who is on call does not answer this line immediately, a voice message can be left and the call will be returned shortly. This team of highly trained staff is available to react to any problem and to do whatever is necessary to ensure customer satisfaction.

Technical Support:

- 1-613-686-1557
- 1-833-859-0499 (Toll free within North America)
- +800 3540 3545 (Toll free International)
- 1300 007 677 (Australia/Sydney)*
- E-mail: techsupport@rossvideo.com
- Website: http://www.rossvideo.com

*If the local support specialist is not available, your call will be transferred automatically to our North America center.

Installation Notes

Voyager 8.0 is compatible with Windows 10, build 1809 and forward and Lucid 8.0.

It's best not to install Voyager and the Unreal Engine on the same computer, as you will have to re-install Voyager to launch projects properly. If you do want both on the same computer, install the Unreal Engine first and then Voyager. It's also recommended not to run both applications at the same time, to avoid compatibility issues.

Dependencies

Voyager engine requires the following:

- .NET Framework 3.5. If .NET Framework 3.5 is not installed, Voyager will prompt the user to download and install this feature. This requires an internet connection.
- Microsoft® Visual C++ 2015-2022 Redistributable 14.40.33810 (or higher)

Switchboard Launcher requires some dependencies that are automatically downloaded and installed when executed the first time. This requires an internet connection.

NDI® Plugin

The NDI plugin included in Voyager 8.0 requires that NDI 6 or newer be installed.

NDI® is a registered trademark of Vizrt ND AB.

Voyager Designer License

With a Voyager Designer license you can use the Voyager editor without camera tracking. Input/Output functionality is enabled, but the output will be watermarked. In the editor, there is one watermark located in the Viewport. In **Play In Editor** or **Game** mode, there are 9 static watermarks in the output, located at the center, top, bottom, and left and right corners.

Voyager NDI® Streams License

To use the NDI plugin for NDI inputs and outputs in Voyager 8.0, you will need the Voyager NDI Streams license on your Voyager system.

The Voyager NDI Streams license includes a specified number of input/output streams (the Count).

Licensed Features:

Feature	Count	Purchase Date	Software Maintenance Expiry	Eligible SW Version
XPression Studio	1	2023-11-28	2024-11-28	At least 11.x
XPression Studio Flex	1	2024-02-01	2025-02-01	At least 11.x
XPression BlueBox SCE	1	2023-11-28	2024-11-28	At least 11.x
XPression BlueBox	1	2023-11-28	2024-11-28	At least 11.x
Lucid Data	1	2023-11-28	2024-11-28	
Lucid Studio	1	2023-11-28	2024-11-28	
Lucid Track	3	2023-11-28	2024-11-28	
Voyager	1	2023-11-28	2024-11-28	
Voyager DataLinq Plugin	1	2024-02-01	2025-02-01	
Voyager Lucid Plugin	1	2024-02-01	2025-02-01	
Voyager NDI Streams	2	2024-07-30	2025-07-30	
Voyager nDisplay Controller	1	2024-02-01	2025-02-01	
Voyager nDisplay Node	1	2024-02-01	2025-02-01	
Voyager Raiden Plugin	1	2024-02-01	2025-02-01	
Voyager RossTalk Plugin	1	2024-02-01	2025-02-01	
Voyager Tracker Plugin	1	2024-02-01	2025-02-01	
Voyager Trackless Plugin	1	2024-02-01	2025-02-01	
XPression Chromakeyer Option	1	2024-07-25	2025-07-25	At least 11.x
XPression DataLinq Support	1	2024-07-25	2025-07-25	At least 11.x

Voyager NDI Streams License

For example, if your Voyager system has a Voyager NDI Streams license with a Count of "2", it can use a maximum of 2 NDI Streams, in any combination:

- 1 NDI Input and 1 NDI Output
- 2 NDI Inputs
- 2 NDI Outputs

Driver Recommendations

It is recommended that the following drivers be installed:

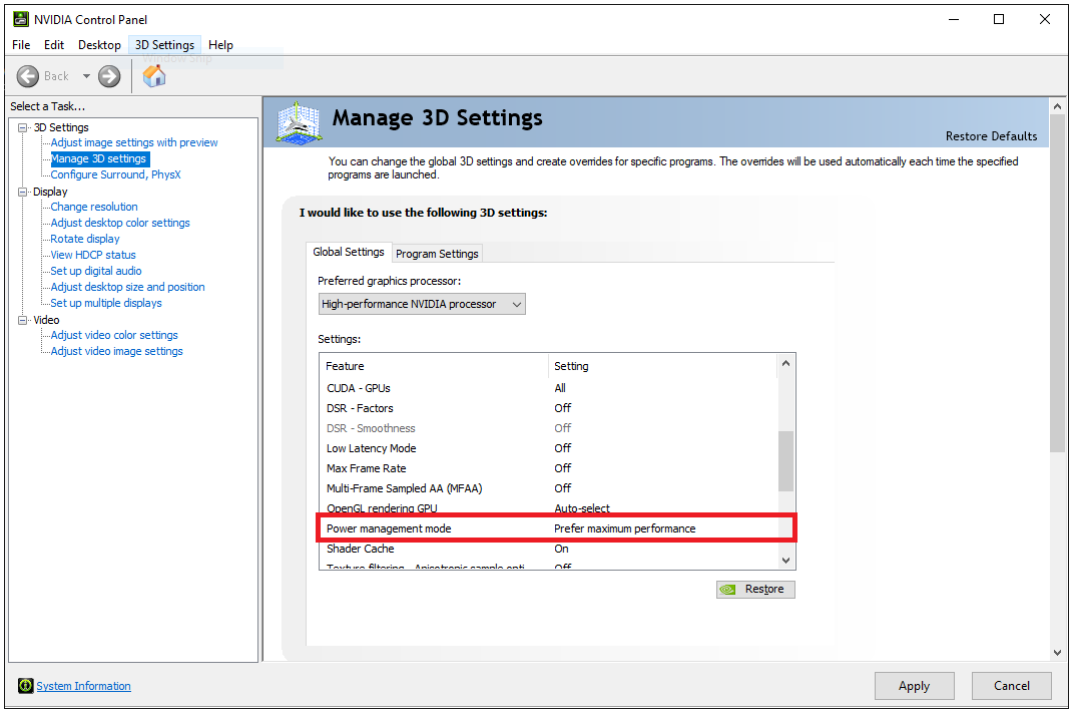
- nVidia Driver 581.80
- Matrox Driver v10.4.101.1334

nVidia Power Management Settings

If your Voyager engine contains the nVidia Quadro Sync II card, you will need to adjust the nVidia power management settings.

To adjust nVidia power management settings:

1. In the **nVidia Control Panel**, select **Manage 3D Settings**.



nVidia Power Management Settings

2. In the **Global Settings** tab, in the **Settings** section, scroll down to **Power management mode** and from the drop-down, select **Prefer maximum performance**.
3. Select **Apply** and close the control panel.

Multi-Engine Systems

When working in a multi-engine configuration, some rules need to be followed:

- Each engine runs its own exact copy of the projects. Those projects needs to be copied in the exact same local path on every engine. **Example: D:\Voyager_Projects**
- The Voyager software itself also needs to be installed in the same exact local path on every engine. By default, Voyager software is installed to **C:\Program Files\Voyager**.

Getting Started

When creating a Voyager project, you'll need to consider whether your project will use internal or external compositing and whether it will be a Virtual Set (VS) or Augmented Reality (AR) project. The difference in settings between VS and AR will be explained within the instructions for internal and external compositing.

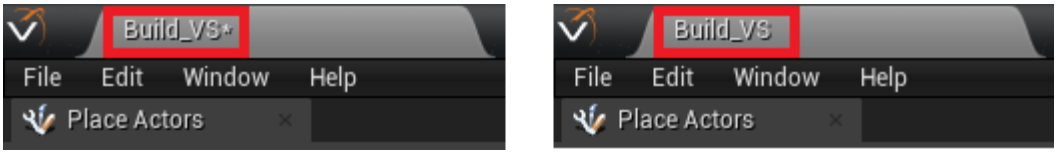
The instructions in this guide reference a system that has a Matrox video card. If your Voyager system uses an AJA video card, anywhere in the instructions where it says to select "Ross Matrox...", select the AJA version of that component instead.

You can choose one of the following methods to set up your project:

Creating a Project from a Template

Making an Existing Project Compatible with Voyager

★When you are working in Voyager and haven't yet saved your work, an asterisk appears beside the title of the window in which you are working. When you save your work, the asterisk disappears.



Before and After Saving

Selecting a UI Layout

Voyager has been updated with the latest version of the Unreal Engine, which resulted in many changes to the UI. If you would like to continue using the classic UE4 layout, you can elect to do so. Otherwise, the same functionality remains, but you may have to look in different locations to find a particular setting. In this document, the screenshots and instructions are for the classic layout.

To use the classic UE4 layout:

1. Launch Voyager and select the project or template you want to use.
2. Then select **Window > Load Layout > UE4 Classic Layout**.

Creating a Project from a Template

Voyager provides several templates to make setting up your project easier. This is the recommended method for creating projects as many of the required assets and settings are included in the templates. You can still import files from an existing project, if you have one prepared that you want to use.

There are a number of steps that are the same for all types of projects, except the Virtual LED project.

The Virtual LED project requires no media proxies, media profile or media bundle.

The rest of the project types require all of the following steps to create:

Setting up a Project

Creating a Media Source Proxy (not necessary for a Virtual LED project)

Select or creating a Media Profile (not necessary for a Virtual LED project)

Configuring the Voyager Operator

Selecting the Startup Media Profile

Creating a Media Bundle (not necessary for a Virtual LED project)

Multiple Composites

If you want more composite inputs in your project than the one that is included with the template, see:

Using Multiple Composite Inputs in Your Project

Live Source Materials

For instructions on creating a material for a live source, see Creating a Live Source Material

Template-Specific Instructions

The following templates require some additional setup. Once you've completed the above steps, go to the section corresponding to your selected template for further template-specific instructions.

Augmented Reality + Virtual Set Template with Set Extension

Virtual LED Template

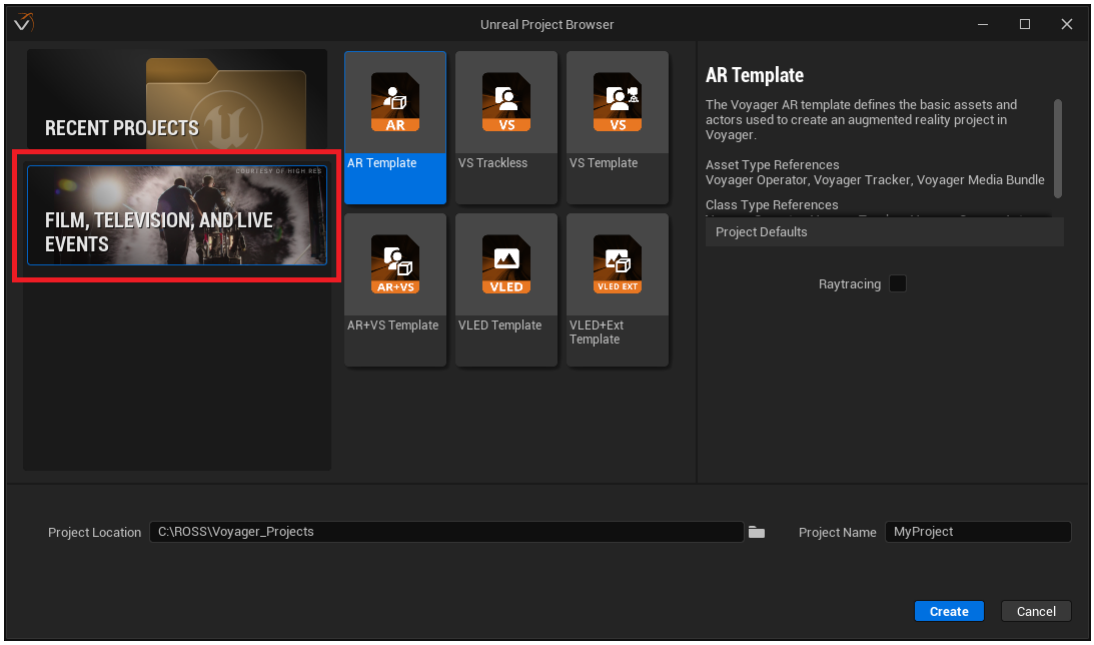
Virtual LED + Set Extension Template

Setting up a Project

The first step is to create an empty project from the template. Then you can migrate files from an existing project to the template and add the migrated level(s) to the new project.

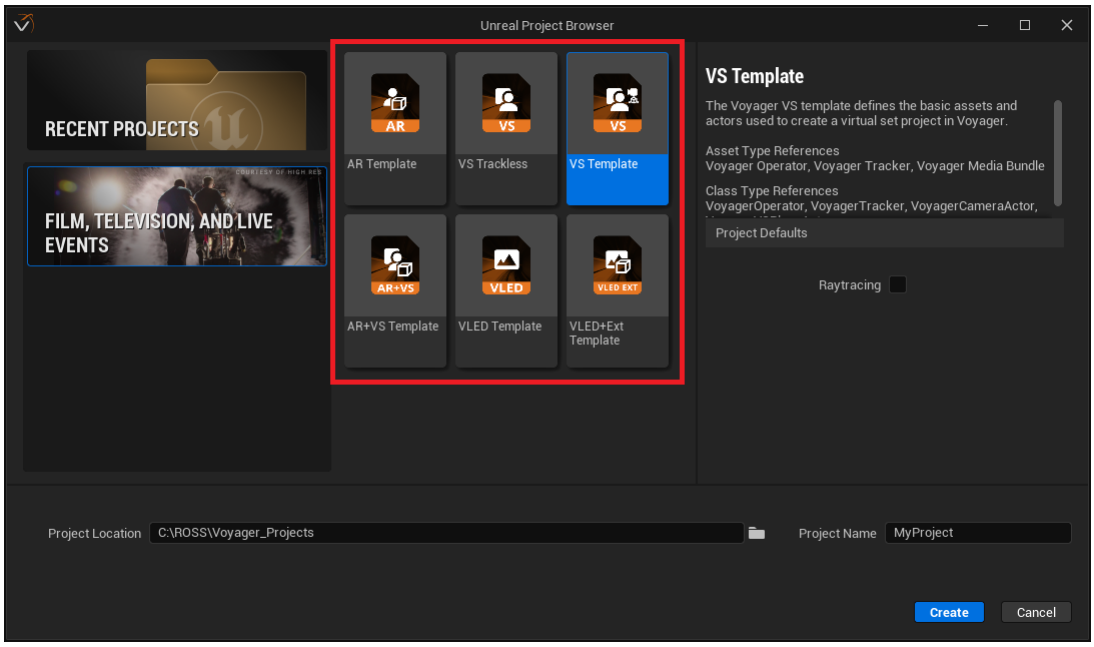
To create an empty project:

1. Launch Voyager from the desktop icon.
2. In the **New Project Catagories** section, select the **Film, Television and Live Events** option and select **Next**.



Select Film, Television and Live Events

3. In the **Select Template** window, select the template you want to use and select **Next**.



Select Template

4. In the **Project Settings** window, from the **Raytracing** drop-down, select whether or not to use real-time raytracing in your project.

   If you don't know which option to choose, select **Raytracing Disabled**. You can change this setting later in the project settings. Raytracing can be very GPU-intensive, so only use it if necessary.

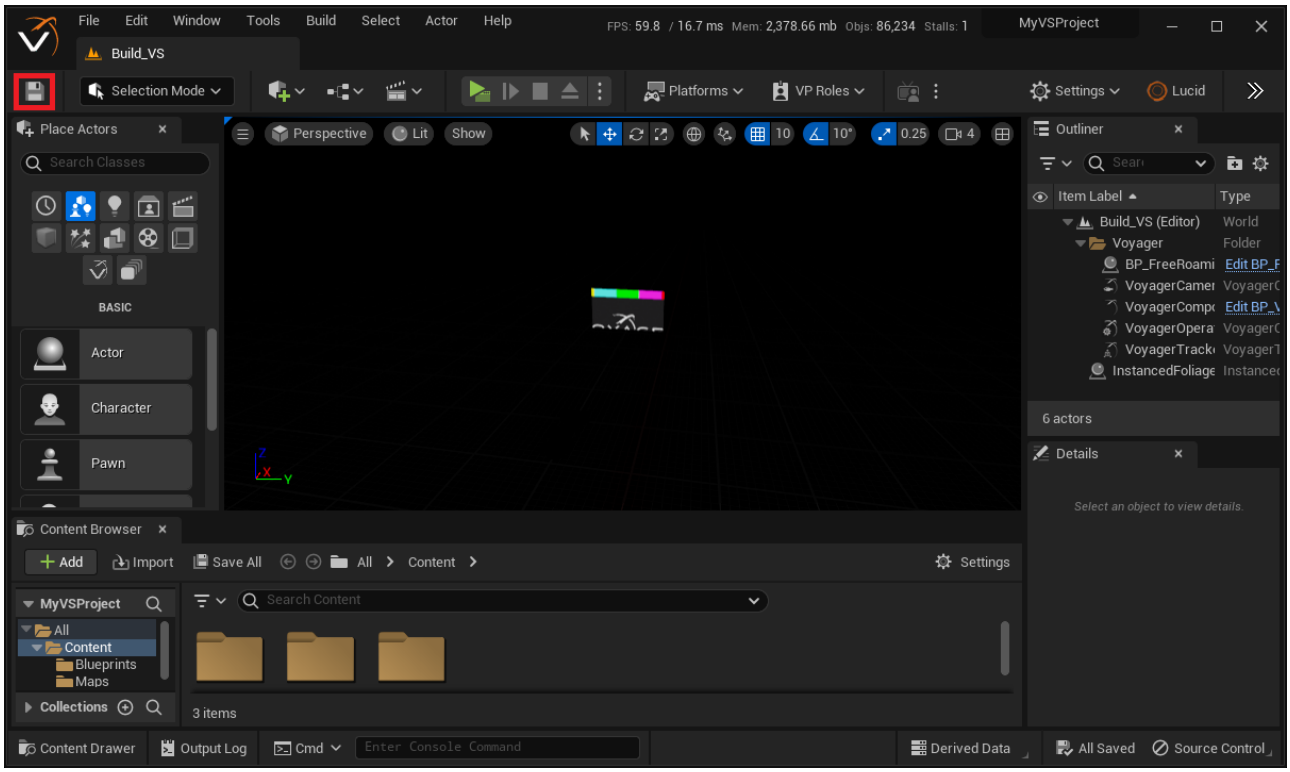
5. Select **Browse** beside the **Project Location** field to navigate to a folder in which to save your new project (e.g., D:\Voyager_Projects) and select **Select Folder**.

6. In the **Project Name** field, enter a name for your project.

   Your project name cannot have any spaces in it. Use underscores rather than spaces.

7. Then select **Create**.

   Your new project opens.



New Project from Template

8. Select the **Save Current** icon and minimize your empty project to the task bar.

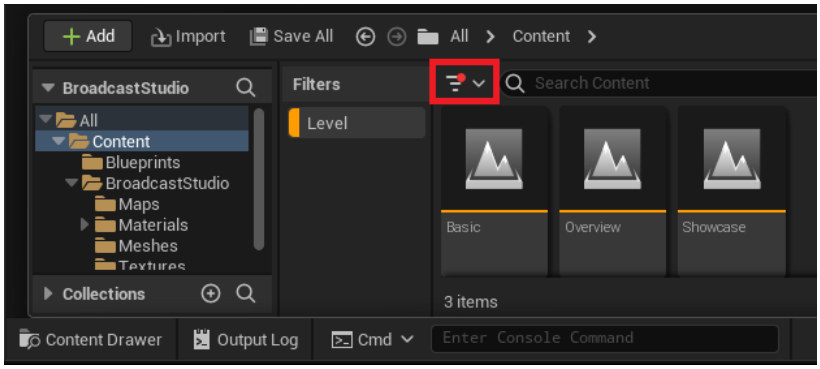
Migrating Files into a Project

Next you'll need to migrate the levels you want to use from an existing project into the template.

To migrate files into a project:

1. Launch the source project containing the level(s) you want to use, by double-clicking the **.uproject** file.
2. In the **Select Unreal Engine Version** dialog, select the latest version of the Unreal Engine and select **OK**.
3. In the **Multiple Instances Running** dialog, select **Yes**.

   Wait a few minutes for the project to launch and the shaders to compile.
4. In the **Content Browser**, select the **Add Filter** arrow and select **Level** from the **Filter** menu to view the project levels.

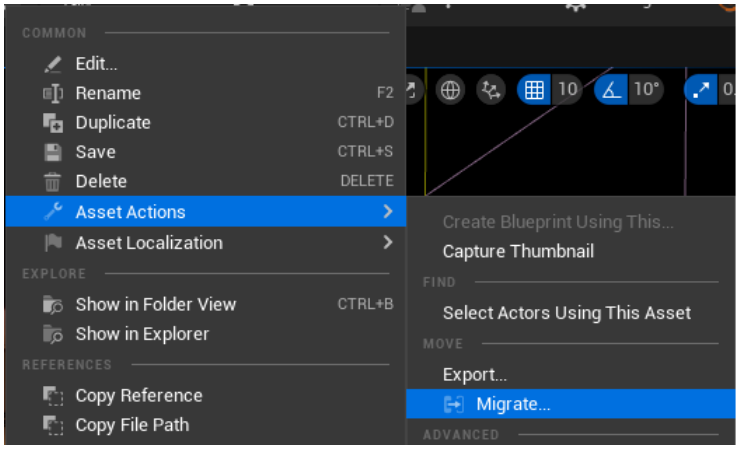


Select Level Filter

5. Double-click the level you want to use to load it in the viewport, if it is not already displayed.
6. In the **Content Browser**, right-click the level and select **Asset Actions > Migrate**.

   You can migrate more than one level at a time, by using **Ctrl + Click or Shift + Click** to select multiple levels.

   ★You can only migrate to the same version of the Unreal Engine or forwards. You cannot migrate backwards from a newer version.



Migrate Level

7. In the **Asset Report** screen, select **OK**.

8. In the **Choose a destination Content folder** screen, navigate to the **Content** folder of your new project and select **Select Folder**.

   The level files are copied to the new project and you'll see a "Content migration completed successfully!" message when it is complete.

9. Close the source project and restore your new project.

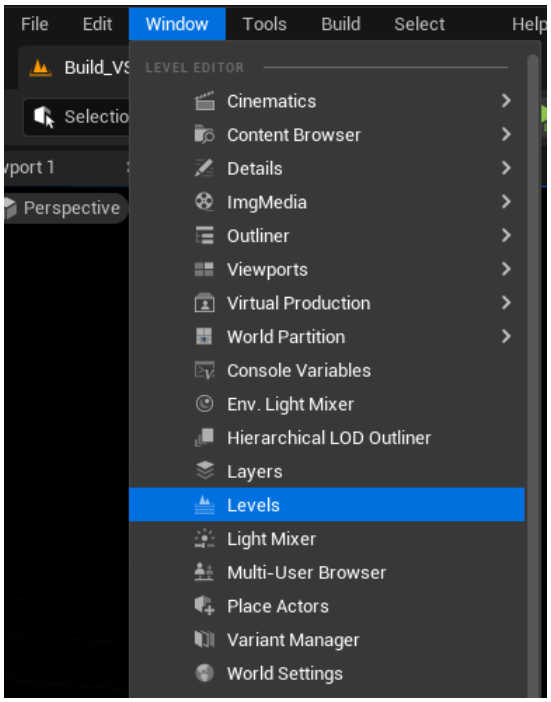
   You'll see that in addition to the default folder, the **Content Browser** now displays the migrated folder(s).

Adding the Migrated Level(s) to the Project

Now you'll add the migrated level to your new project.

To add migrated levels to the project:

1. In the new project, select **Window > Levels** to open the **Levels** panel if it is not already present.



Add Levels Panel

2. Select and drag the **Levels** panel tab to position it where you want in the UI.
3. In the **Content Drawer**, select **Filters > Level** to display the levels in your project.

   You will see the default Voyager level (**Build_"Template Type"**) and the migrated level(s).

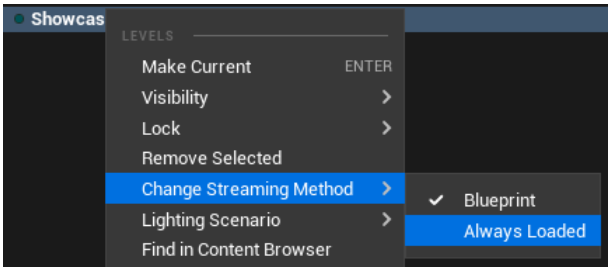
4. Drag the migrated project level into the **Levels** panel, where it will be nested under the **Persistent Level** folder.

   Notice that in the **Levels** panel, the migrated level name is blue and there is a dot beside it. This indicates that a blueprint is required to launch the level.

5. If you only have one level in your project, right-click the level in the **Levels** panel and select **Change Streaming Method > Always Loaded** instead of creating a blueprint.

   The level will be selected and played when the project is launched. Notice that the level name is no longer blue and there is no dot beside it.

   If you have more than one level in your project, see Creating a Blueprint to Launch Multiple Levels.



Change Streaming Method

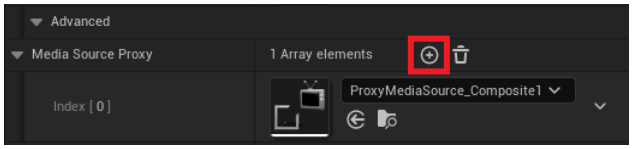
6. Select the **Save** button to save your project.
7. Continue with Creating a Media Source Proxy.

Creating a Media Source Proxy

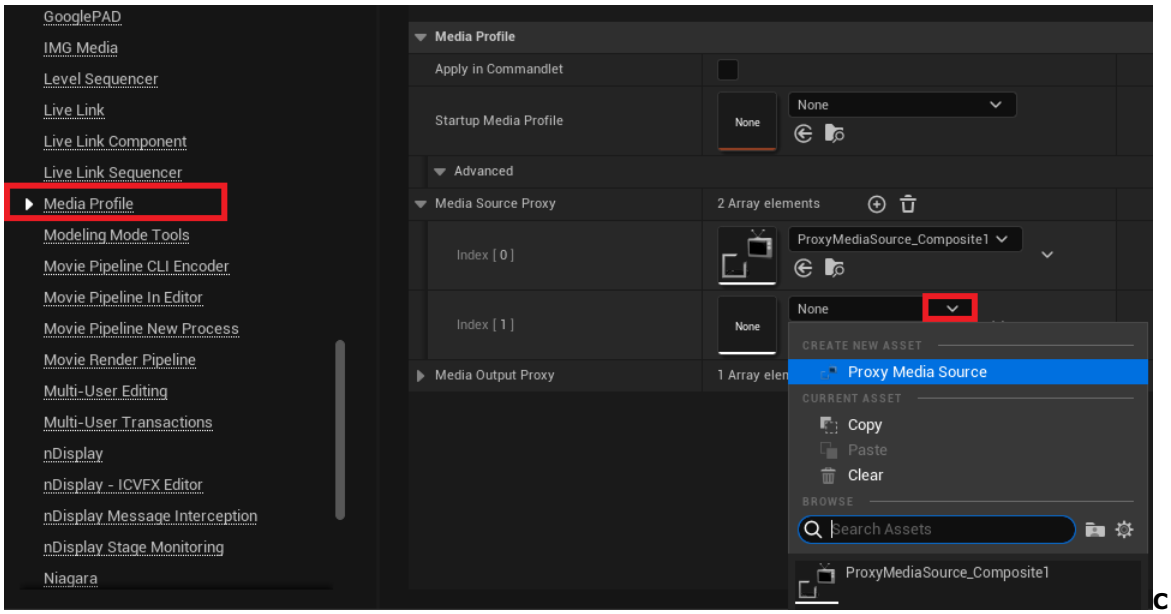
With your basic project created, you now need to create media source proxies for any additional composite input in your project. Voyager templates have 1 **Media Source Proxy** and 1 **Media Output Proxy** already configured.

To create a media source proxy:

1. Select **Edit > Project Settings** and scroll down to the **Plugins** section.
2. From the **Plugins** section, select **Media Profile**.
3. Expand the **Advanced** options.
4. In the **Media Source Proxy** section, select the **+** icon to add an array element for each additional composite plane in your project.



5. Select the arrow in the new array drop-down and select **Create New Asset > Proxy Media Source**.



C

Create a Media Source Proxy

6. In the **Save Asset As** window, navigate to the **Voyager > Proxies** folder and in the **Name** field, enter a name for the media source proxy (e.g. **ProxyMediaSource_Composite2**).
7. Select **Save**.
8. Repeat steps 5 to 7 so that you have a media source proxy for each composite plane in your project.
9. When you have finished, close the **Project Settings** window and continue with Selecting or Creating a Media Profile.

Selecting or Creating a Media Profile

If you do not have a media profile on your engine, you'll need to create one. The media profile defines the number of inputs being used and configures the inputs and output for your project.

★You will not need a media profile for a Virtual LED project.

Selecting an Existing Media Profile

If you have previously created a media profile on your engine, it will be saved in **Engine Content** and is available for use with any project.

To select an existing media profile:

- Select the arrow beside the **Media Profile** icon, and select the media profile that matches your requirements.

Accessing the Media Profiles Folder

If you need to create a media profile, you'll need to access a folder in the **Engine Content**. It is best not to have the core engine files visible all the time, as inadvertently changing something in these files could interfere with your Voyager installation. Make them visible only while it is necessary, then hide them again.

To view engine content:

1. In the bottom-right corner of the **Content** space, select **View Options** and select the **Show Engine Content** checkbox.
2. Proceed with the instructions for creating a media profile.
3. When you have finished creating your media profile, go back to **View Options** and deselect the **Show Engine Content** checkbox.
4. If you want to reuse the media profile you create here in another project, you'll need to again go to **View Options**, select the **Show Engine Content** checkbox and in the **Content Browser** navigate to the profile in the **MediaProfiles** folder.

Creating a Media Profile

The steps for creating a media profile are described in the following sections:

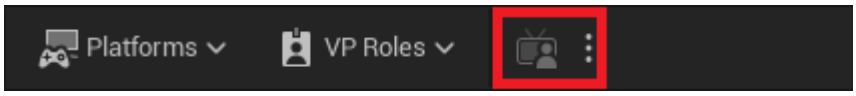
Creating an Empty Media Profile

This step creates the container that stores the input and output configurations.

You will not need a media profile for a Virtual LED project.

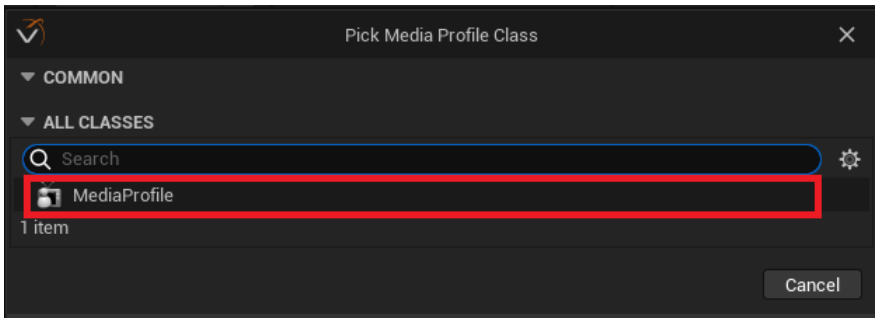
To create an empty media profile:

1. In the main toolbar, select the 3 vertical dots beside the **Media Profile** icon.



Media Profile Icon

2. From the drop-down menu, select **New Empty Media Profile**.
3. In the **Pick Media Profile Class** section, select the **MediaProfile** item and select **Select**.

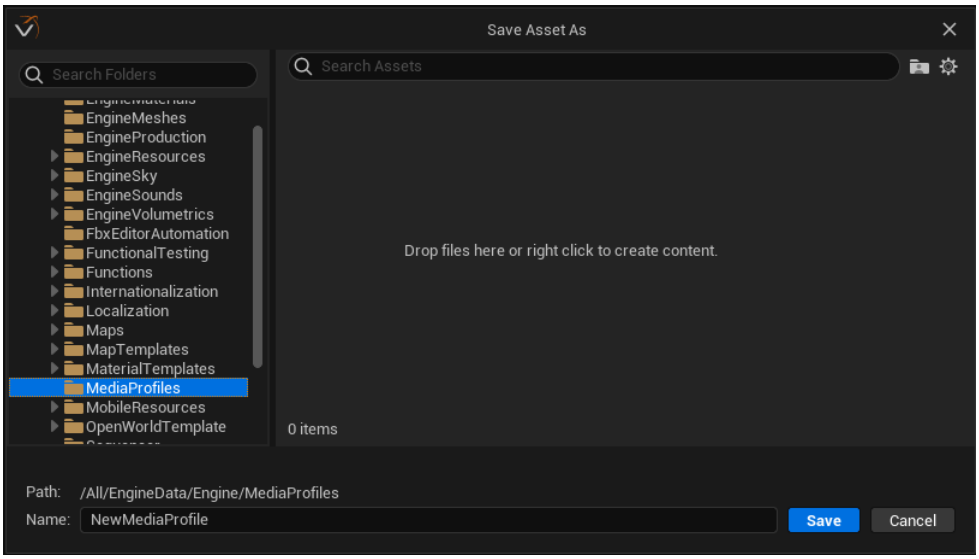


Pick Media Profile Class

4. In the **Save Asset As** window, select the settings icon to the right of the **Search Assets** field and select **Show Engine Content**.

5. Then in the **Search Folders** list, select **Content > Engine > Content**, scroll down and select the **MediaProfiles** folder.

   - If the **MediaProfiles** folder does not exist, right-click **Engine > Content** and select **New Folder** to create a folder named **MediaProfiles**.



Save Asset As

6. In the **Name** field at the bottom of the **Save Asset As** window, enter a name for the media profile you will create and select **Save**.

   e.g., **Matrox_1080p_5994_VS**.

   The new empty media profile is saved in the **MediaProfiles** folder and the **Details** tab opens.

7. Continue with Configuring Inputs.

Configuring Inputs

Now you'll configure your input(s). The instructions in this section describe how to configure a composite input. If you want to create a live source input, see Creating Live Sources. You'll also need to create a live source material; see Creating a Live Source Material.

Select your hardware version for the appropriate instructions.

Voyager SDI and 12G

Voyager IP

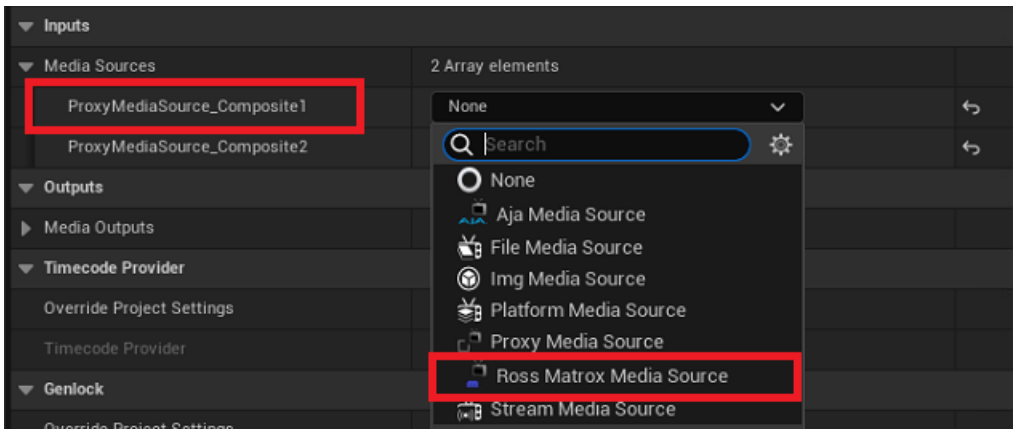
Voyager SDI and 12G

The configuration instructions in this section are for configuring a composite input for the Voyager SDI and 12G versions.

If you want to configure an input for a live source, see Creating Live Sources. You will also need to create a live source material for each live source; see Creating a Live Source Material.

To configure a composite input:

1. Double-click the **Media Profile** icon in the main tool bar to open the media profile you created, if it is not already open.
2. Expand the **Inputs > Media Sources** section, then select the **ProxyMediaSource_Composite1** drop-down and select **Ross Matrox Media Source**.



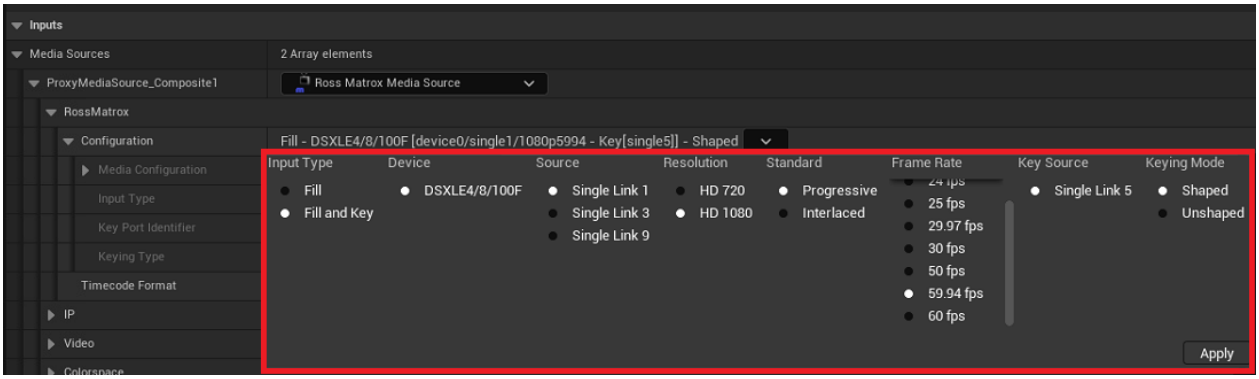
Media Source Input

3. Expand the **ProxyMediaSource_Composite1** input and then expand **RossMatrox** to access **Configuration**.

4. Select the **Configuration** drop-down and in the configuration panel that opens, select the options described below and select **Apply**.

   The **Resolution** and **Frame Rate** options will differ depending on your hardware configuration.

   - **Input Type**: Fill and Key (for a VS with an external keyer) or Fill (for VS with an internal keyer or AR)
   - **Device**: DSXLE4/8/100F
   - **Source**: Single Link 1 (first composite), Single Link 3 (2nd composite), etc.
   - **Resolution/Standard/Frame Rate**: The video format that corresponds to your workflow.
   - **Key Source**: The appropriate key source will be automatically selected, based on your selection of **Source**, if using a Matrox card. If using an AJA card, select the appropriate key source manually.
   - **Keying Mode**: Select **Shaped** (pre-multiplied) or **Unshaped** (straight alpha).



   Input Configuration - Composite Input

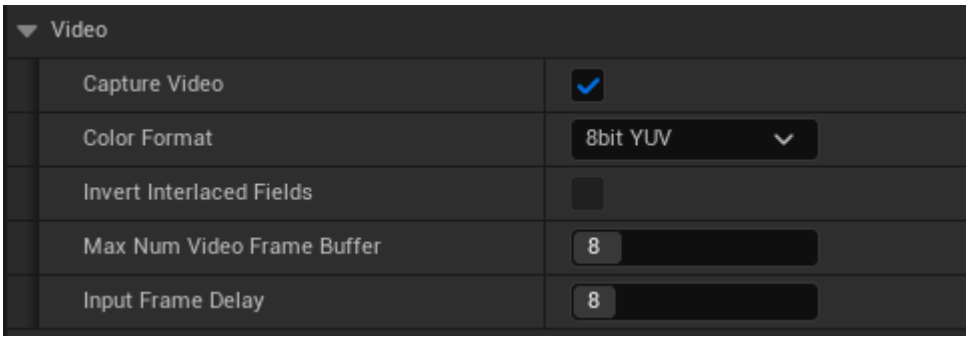
5. If you intend to use the **Timecode Provider**, from the **Timecode Format** drop-down, select **VITC**.

6. Expand **Video** and configure the settings as follows:
   - Select the **Capture Video** checkbox.
   - From the **Color Format** drop-down, select the color format that works best for your project.

     10bit YUV 422 is recommended when using **HDR** and **Wide Color Gamut**.

     With multiple inputs, there may be a performance cost when using **10bit YUV 422**. In this case, try using **8bit YUV 422**.

     10bit YUV 422 is not supported in a **Fill and Key** configuration with an AJA card. In this case you could use **RGBA**.
   - If you have an AJA card with the **Interlaced Standard**, and you notice field inversion, select the **Invert Interlaced Fields** checkbox.
   - From the **Max Num Video Frame Buffer** drop-down, select **8** if you are using a **Progressive** format or if you are using an **Interlaced** format, select **16**.
   - From the **Input Frame Delay** drop-down, select the frame delay that works best for your project.



Video Configuration - Composite Input

7. Expand **Colorspace** and configure the settings as follows:

   - From the **Colorimetry** drop-down, select one of the following options:

     ➢ **Rec. 709 (HD SDR)** for High Dynamic Range (increased levels in the range between bright and dark) and Standard Dynamic Range

     OR

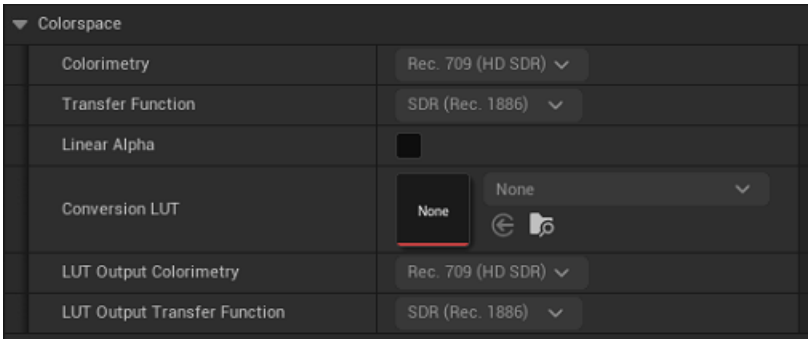
     ➢ Rec. 2020 (WCG) for Wide Color Gamut (increased selection of color values)

   - From the **Transfer Function** drop-down, select one of the following options:

     ➢ **SDR (Rec. 1886)** — Standard Dynamic Range

     ➢ **HLG (Rec. 2100)** — increases the dynamic range of the video and is compatible with both SDR and HDR displays

     ➢ **HDR10 (PQ 1000 nits)** — supports a significantly larger range of brightness as SDR, with a corresponding increase in contrast and a color palette of one billion shades.



Colorspace Settings

   - Select the **Linear Alpha** checkbox if the incoming alpha is already linear; the **Transfer Function** will not be applied.

     Refer to the documentation for your chroma keyer or key source to determine whether or not the alpha is linear.

   - If you selected an 8bit color format in the **Video** settings, these are the only **Colorspace** settings available. Select **Save** and continue with the output configuration.

   - If you selected a 10bit color format in the **Video** settings, the **HDR** settings will also be available. Continue with the next steps to edit **HDR** settings.

     ➢ From the **Conversion LUT** drop-down, browse to and select the **Look Up Table** you want to apply to your color grading.

     ➢ From the **LUT Output Colorimetry** drop-down, select either **Rec. 709 (HD SDR)** or **Rec. 2020 (WCG)**.

     ➢ From the **LUT Output Transfer Function** drop-down, select either **SDR (Rec. 1886)**, **HLG (Rec. 2100)** or **HDR10 (PQ 1000 nits)**.

8. Select **Save**.

Continue with Configuring an Output.

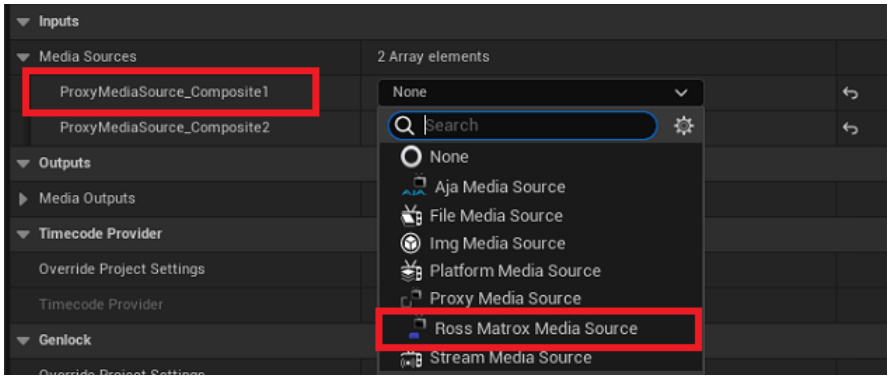
Voyager IP

The configuration instructions in this section are for the Voyager IP version.

If you want to configure an input for a live source, see Creating Live Sources. You will also need to create a live source material for each live source; see Creating a Live Source Material.

To configure a composite input:

1. Double-click the **Media Profile** icon in the main toolbar to open the media profile you created.
2. Expand the **Inputs > Media Sources** section, then select the **ProxyMediaSource_Composite1** drop-down and select **Ross Matrox Media Source**.

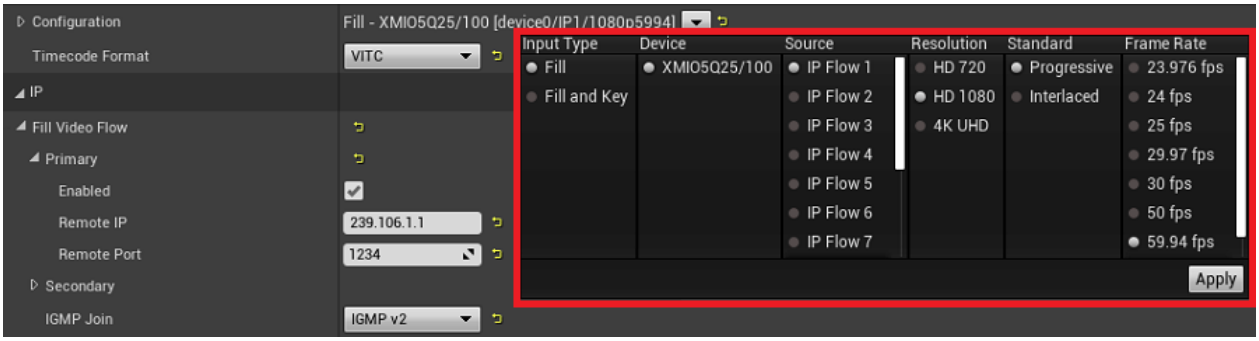


Media Source Input

3. Expand the **ProxyMediaSource_Composite1** input and then expand **RossMatrox** to access **Configuration**.
4. If you want to use the **Timecode** functionality, from the **Timecode Format** drop-down, select **VITC**.
5. Select the **Configuration** drop-down and in the configuration panel that opens, select the options described below and select **Apply**.

   The **Resolution** and **Frame Rate** options will differ depending on your hardware configuration.

   - **Input Type**: Fill and Key (for a VS with an external keyer) or Fill (for VS with an internal keyer or AR)
   - **Device**: Matrox
   - **Source**: IP Flow 1
   - **Resolution/Standard/Frame Rate**: The video format that corresponds to your workflow.



Input Configuration - Composite Input (IP)

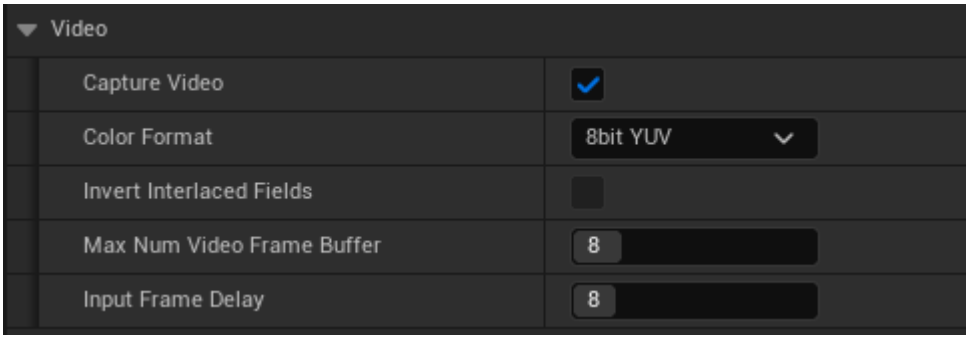
6. Expand the **IP** section and then expand **Primary** and configure the settings as follows:
   - Select the **Enabled** checkbox.
   - In the **Remote IP** field, enter the IP address of the remote machine.
   - In the Port field, enter the port number on which the machines will be communicating.
   - Expand **Secondary** and from the **IGMP Join** drop-down, select **IGMP v2**.
7. If you are using a **Matrox IP 2110** card and require timecode functionality, expand **Ancillary Flow** and **Primary** and configure the settings as follows:
   - Select the **Enabled** checkbox.
   - In the **Remote IP** field, enter the IP address of the remote machine.
   - In the **Port** field, enter the port number on which the machines will be communicating.
   - Expand **Secondary** and from the **IGMP Join** drop-down, select IGMP v2.
8. Expand **Video** and configure the settings as follows:
   - Select the **Capture Video** checkbox.
   - From the **Color Format** drop-down, select the color format that works best for your project.

     10bit YUV 422 is recommended when using **HDR** and **Wide Color Gamut**.

     With multiple inputs, there may be a performance cost when using **10bit YUV 422**. In this case, try using **8bit YUV 422**.

     10bit YUV 422 is not supported in a Fill and Key configuration with an AJA card. In this case you could use **RGBA**.

     Leave the remaining settings as they are.
   - If you have an AJA card with the **Interlaced Standard**, and you notice field inversion, select the **Invert Interlaced Fields** checkbox.
   - From the **Max Num Video Frame Buffer** drop-down, select **8** if you are using a **Progressive** format or if you are using an Interlaced format, select **16**.
   - From the **Input Frame Delay** drop-down, select the frame delay that works best for your project.



Video Configuration - Composite Input

9. Expand **Colorspace** and configure the settings as follows:

   - From the **Colorimetry** drop-down, select one of the following options:

     ➢ **Rec. 709 (HD SDR)** for High Dynamic Range (increased levels in the range between bright and dark) and Standard Dynamic Range

     OR

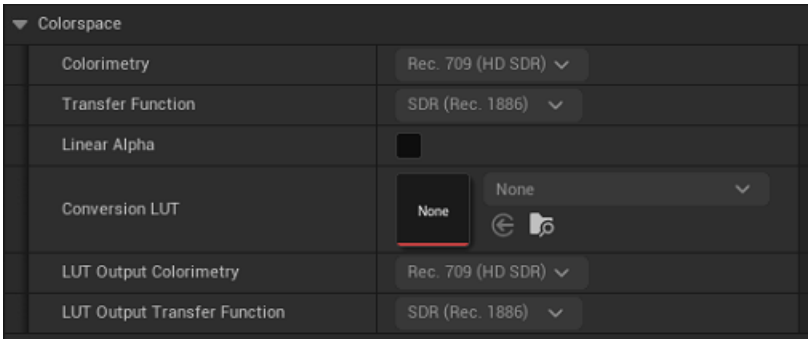
     ➢ Rec. 2020 (WCG) for Wide Color Gamut (increased selection of color values)

   - From the **Transfer Function** drop-down, select one of the following options:

     ➢ **SDR (Rec. 1886)** — Standard Dynamic Range

     ➢ **HLG (Rec. 2100)** — increases the dynamic range of the video and is compatible with both SDR and HDR displays

     ➢ **HDR10 (PQ 1000 nits)** — supports a significantly larger range of brightness as SDR, with a corresponding increase in contrast and a color palette of one billion shades.



Colorspace Settings

   - Select the **Linear Alpha** checkbox if the incoming alpha is already linear; the **Transfer Function** will not be applied.

     Refer to the documentation for your chroma keyer or key source to determine whether or not the alpha is linear.

   - If you selected an 8bit color format in the **Video** settings, these are the only **Colorspace** settings available. Select **Save** and continue with the output configuration.

   - If you selected a 10bit color format in the **Video** settings, the **HDR** settings will also be available. Continue with the next steps to edit **HDR** settings.

     ➢ From the **Conversion LUT** drop-down, browse to and select the **Look Up Table** you want to apply to your color grading.

     ➢ From the **LUT Output Colorimetry** drop-down, select either **Rec. 709 (HD SDR)** or **Rec. 2020 (WCG)**.

     ➢ From the **LUT Output Transfer Function** drop-down, select either **SDR (Rec. 1886)**, **HLG (Rec. 2100)** or **HDR10 (PQ 1000 nits)**.

10. Select **Save**.

    Continue with Configuring an Output.

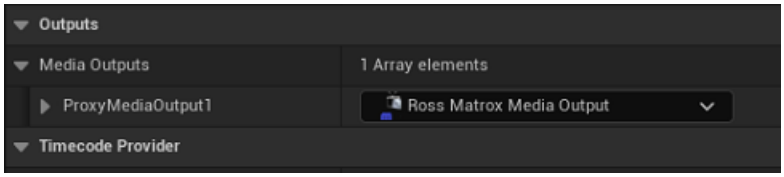
Configuring an Output

There is only one output available in Voyager. You only need to decide whether you want to configure that output for internal compositing or external compositing. Instructions for both are provided in this section.

Internal Compositing

To configure an output for internal compositing:

1. In the **Outputs** section, expand **Media Outputs**.
2. Then select the **ProxyMediaOutput1** drop-down and select **Ross Matrox Media Output**.

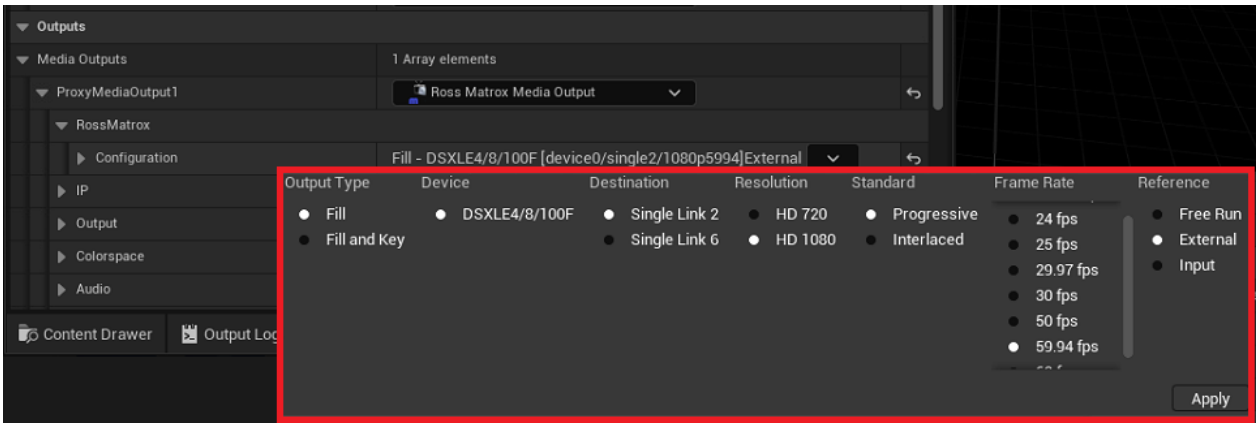


Media Profile - Output Config

3. Expand **ProxyMediaOutput1** and then expand **RossMatrox** to access **Configuration**.
4. Select the **Configuration** drop-down and in the configuration panel that opens, select the options described below and select **Apply**.

   The **Source**, **Resolution** and **Frame Rate** options will differ depending on your hardware configuration.

   - **Output Type**: Fill
   - **Device**: DSXLE4/8/100F
   - **Destination**: The physical pin you are using for your output (different cards will display different options here)
   - **Resolution/Standard/Frame Rate**: The video format that corresponds to your workflow.
   - **Reference**: External



Output Configuration

5. Expand **Output**, select the arrow in the **Pixel Format** field and select the format that works best for your project.

   10bit YUV 422 is recommended when using **HDR** and **Wide Color Gamut**.

   With multiple inputs, there may be a performance cost when using **10bit YUV 422**. In this case, try using **8bit YUV 422**.

   10bit YUV 422 is not supported in a **Fill** and **Key** configuration with an AJA card. In this case you could use **RGBA**.

6. Select the **Parallelize Transfers** checkbox to improve performance.

   Output frames are transferred faster, which can make a noticeable difference in 4K projects.

7. Leave the remaining settings as they are and select **Save**.

   Continue with Configuring the Genlock Settings.

External Compositing

To configure an output for external compositing:

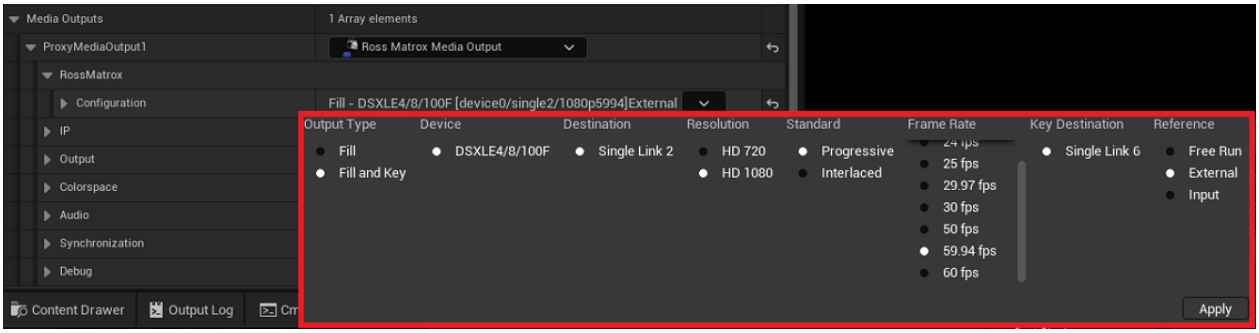
1. In the **Outputs** section, expand **Media Outputs**.
2. Then select the **ProxyMediaOutput1** drop-down and select **Ross Matrox Media Output**.
3. Expand **ProxyMediaOutput1** and then expand **RossMatrox** to access **Configuration**.
4. Select the **Configuration** drop-down and in the configuration panel that opens, select the options described below and select **Apply**.

   The **Resolution** and **Frame Rate** options will differ depending on your hardware configuration.

   - **Output Type**: Fill and Key
   - **Device**: DSXLE4/8/100F
   - **Destination**: The physical pin you are using for your output (different cards will display different options here)

     ★For AJA cards, the **Destination** and **Key Destination** pins need to be adjacent (e.g., **Destination** = Out 1 and **Key Destination** = Out 2).
   - **Resolution/Standard/Frame Rate**: The video format that corresponds to your workflow.
   - **Key Destination**: The appropriate key destination will be automatically selected, based on your selection of **Destination**, if using a Matrox card. If using an AJA card, select the appropriate key destination manually.

     ★For AJA cards, the **Destination** and **Key Destination** pins need to be adjacent (e.g., **Destination** = Out 1 and **Key Destination** = Out 2).
   - **Reference**: External

   Output Configuration - External Compositing
5. Expand **Output**, select the arrow in the **Pixel Format** field and select the format that works best for your project.

   10bit YUV 422 is recommended when using **HDR** and **Wide Color Gamut**.

   With multiple inputs, there may be a performance cost when using **10bit YUV 422**. In this case, try using **8bit YUV 422**.

   10bit YUV 422 is not supported in a **Fill and Key** configuration with an AJA card. In this case you could use **RGBA**.
6. Select the **Parallelize Transfers** checkbox to improve performance.

   Output frames are transferred faster, which can make a noticeable difference in 4K projects.

7. Leave the remaining settings as they are and select **Save**.

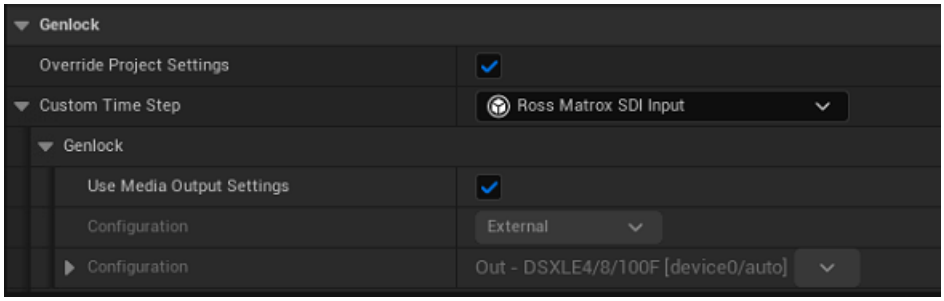
   Continue with Configuring the Genlock Settings.

Configuring the Genlock Settings

Once you have your input(s) and output configured, the last step is to configure the **Genlock** settings. The settings will depend on the type of graphics card you have in your engine. Select the appropriate instructions below:

To configure the Genlock Settings for a Matrox card:

1. In the **Genlock** section, select the **Override Project Settings** checkbox.

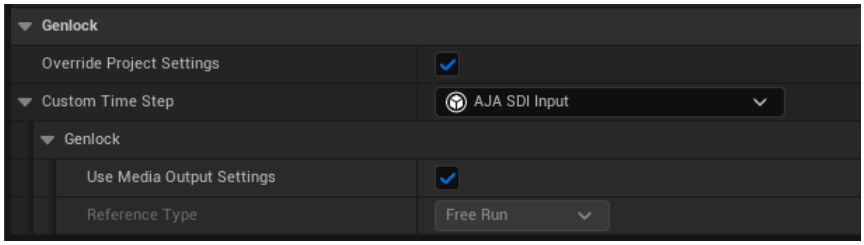


Genlock Configuration - Matrox

2. Select the **Custom Time Step** arrow and from the drop-down select **Ross Matrox SDI Input**.
3. Expand **Custom Time Step** and then expand **Genlock** and select the **Use Media Output Settings** checkbox.
4. Select **Save** and close the **Media Profile** window.

To configure the Genlock Settings for an AJA card:

1. In the **Genlock** section, select the **Override Project Settings** checkbox.



Genlock Configuration - AJA

2. Select the **Custom Time Step** arrow and from the drop-down select **AJA SDI Input**.
3. Expand **Custom Time Step** and then expand **Genlock** and select the **Use Media Output Settings** checkbox.

   This forces the AJA **Custom Time Step** to use the **Genlock** settings configured on the first **AJA Media Output** of the **Media Profile**.
4. Select **Save** and close the **Media Profile** window.

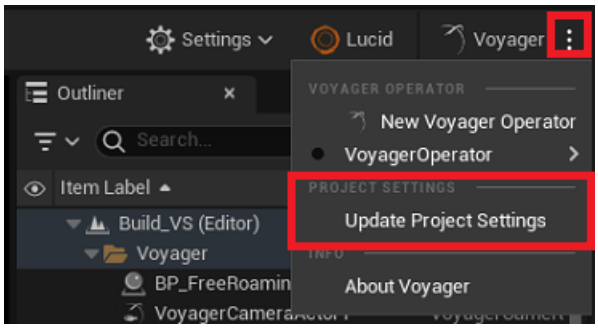
Continue with Updating Voyager Project Settings.

Updating Voyager Project Settings

Once you've set up your project, it's time to update the Voyager project settings.

To update the project settings:

1. Select the 3 vertical dots beside the Voyager icon and select **Select a Voyager Operator > VoyagerOperator**.
2. Then select **Update Project Settings**.



Updating Voyager Project Settings

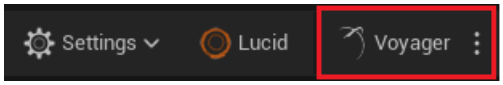
3. In the **Voyager Project Defaults** dialog, select **Yes** to proceed.
4. Continue with Selecting the Startup Media Profile.

Configuring the Voyager Operator

The next step in the process is to configure the Voyager Operator.

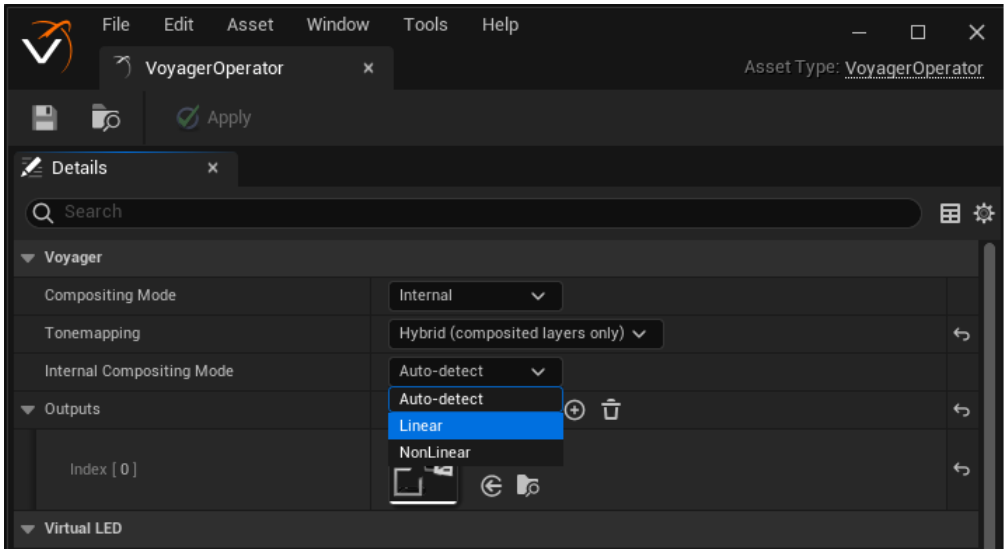
To configure the Voyager Operator:

1. At the top-right corner of the UI, select the **Edit VoyagerOperator** button.



Edit VoyagerEditor Button

The **VoyagerOperator** editor opens.



VoyagerOperator Editor

2. From the **Internal Compositing Mode** drop-down, select from the following options:

   Auto-detect — uses Linear compositing (engine default) unless the Voyager Media Bundle on the first compositing layer is set to **External Keying** and **Tonemapping** is set to **Hybrid**.

   Linear — recommended for multiple compositing layers, as it supports the most features and layering combinations.

   Non-Linear — recommended for a single layer using external keying, as it supports fewer layering and post-processing combinations but replicates the external compositing typical of broadcast devices.

   ★In **Non-Linear** mode, pixels in the virtual graphics that are hidden by composite objects will still generate bloom, as if they were visible. This mode is intended for use with external keyers. It could have undesired effects on the edges of internal chroma keyers.

   ★Additional live sources that are fill-only without using chroma keying can be used in either mode, based on the nature of the main composite's keying method.

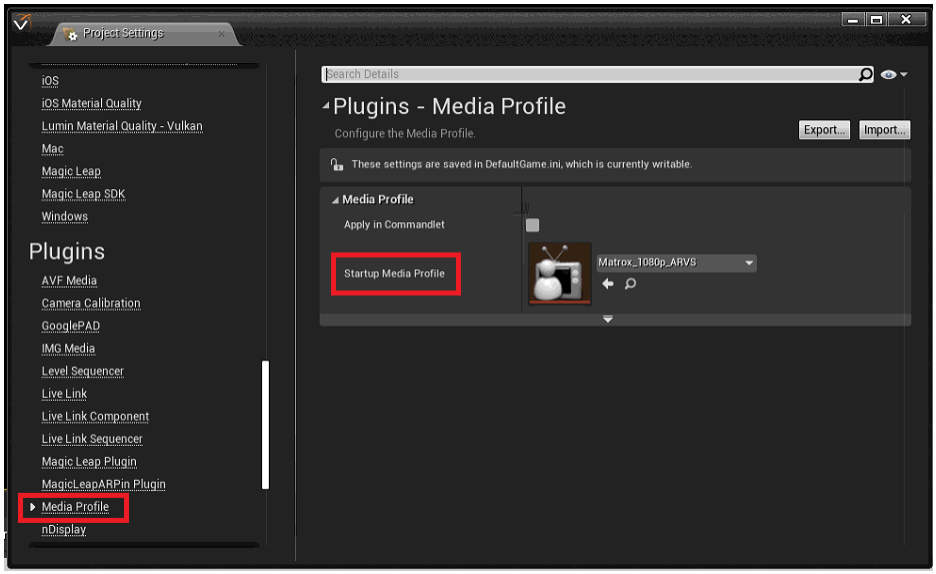
3. Select **Save** and close the VoyagerOperator editor.

Selecting the Startup Media Profile

If you want to run your project in **Game** mode (Voyager Node or nDisplay), you will need to select the **Startup Media Profile**.

To select the Startup Media Profile:

1. Select **Edit > Project Settings > Plugins > Media Profile**.
2. In the **Media Profile** plugin, from the **Startup Media Profile** drop-down, select the media profile you created for your project and then close the Project Settings editor.



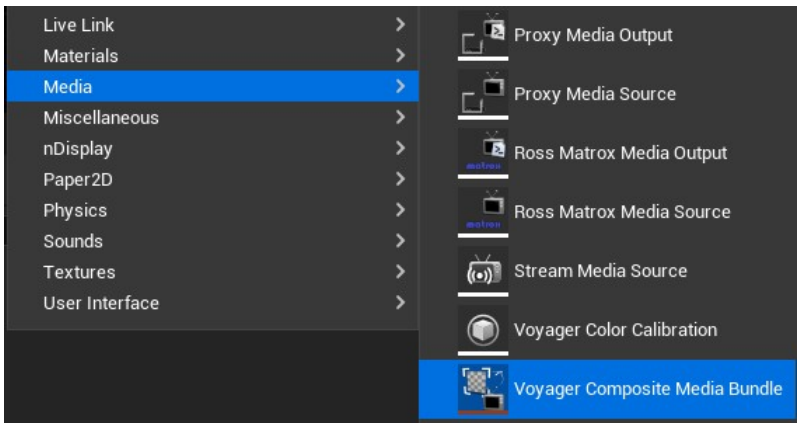
Select Startup Media Profile

Creating a Media Bundle

You need to create a media bundle asset for each input in your project. This will allow the asset to be controlled by other applications.

To create a media bundle:

1. Navigate to the **Voyager > LiveSources** folder.
2. Right-click in an empty section of the **Content** pane and select **Media > Voyager Composite Media Bundle** (for a composite input) or **Media > Media Bundle** (for a live source input).

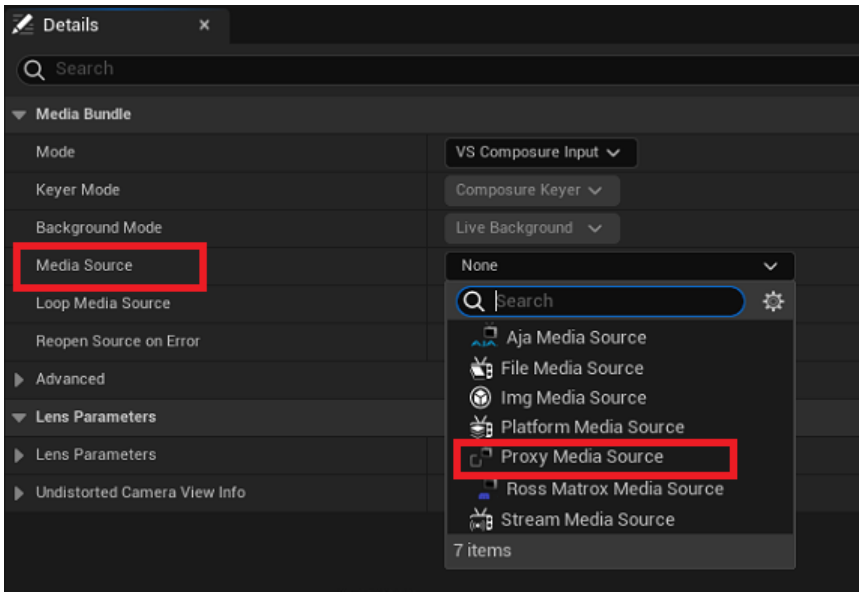


Create VoyagerComposite Media Bundle

3. In the **Content** pane, rename the **Media Bundle** asset to associate it with a composite input (e.g., **VoyagerComposite_1**) or a live source input (e.g., **LiveInput_1**).

   The **InnerAssets** folder is created automatically.

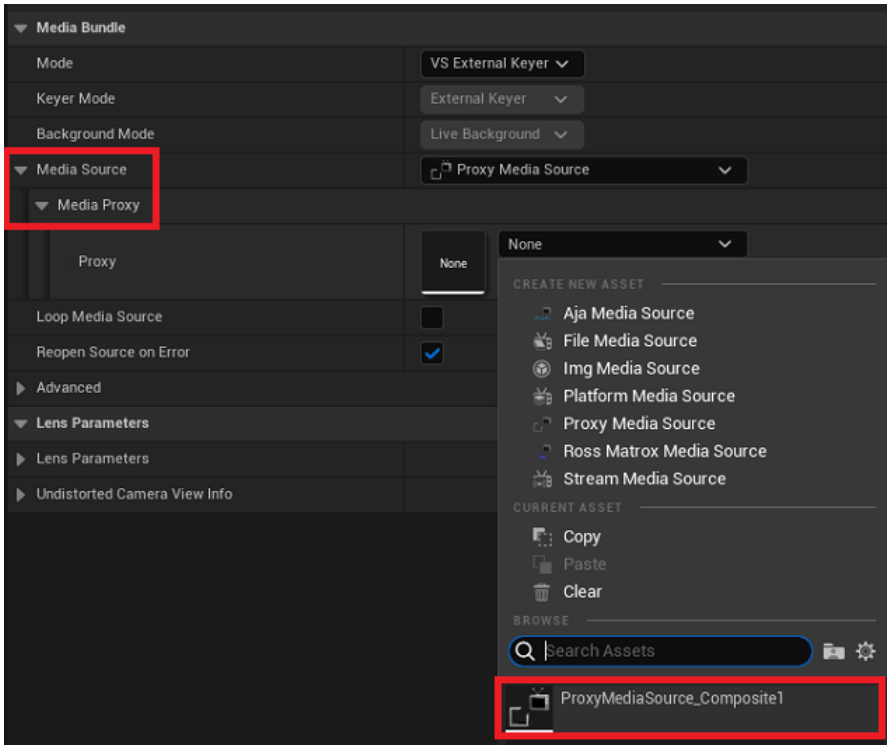
4. Double-click the media bundle you created to open the **Details** tab.
5. From the **Mode** drop-down, select the chroma keyer mode you need for your project.
6. From the **Media Source** drop-down, select **Proxy Media Source**.



Select Media Source

7. Expand **Media Source** and **Media Proxy** and from the drop-down, select the proxy media source to be

used with that media bundle.



Select Proxy Media Source

8. Repeat steps 2 to 7 for each input in your project.
9. Select **Save** and close the **Details** tab.

Augmented Reality + Virtual Set Template with Set Extension

When creating a project that uses both AR and a virtual set with set extension, you may find that you are seeing some reflection from the green screen in parts of the physical set.

You can use the **VoyagerARDespillMask** actor to gray out the green.

By default, the **VoyagerARDespillMask** actor is a plane, but it is a 3D object that can be replaced by a static mesh, if that would be more suitable for masking a particular area.

The **VoyagerARDespillMask** is used in conjunction with the **despill_matte** actor. The **despill_matte** actor determines how the despill will be applied, either to an area in the AR background outside of the **VoyagerARDespillMask** object (**Garbage Matte**) or only to the area defined by the **VoyagerARDespillMask** object (**Holdout**).

You can also feather the edges of the mask in order to provide a more seamless blending of the masked area and the unmasked area.

Using the VoyagerARDespill Mask

If you are seeing some reflection from the green screen in parts of the physical set, you can mask out the reflection with the **VoyagerARDespill Mask**.

To use the VoyagerARDespillMask:

1. Determine to which part of the set you want to apply the mask.
2. In the **Outliner**, select the **despill_matte** actor.
3. In the **Details** tab, scroll down to **Composure > Input > Matte Type** and select the appropriate option:
   - **Garbage Matte** - to remove any green reflection from the camera feed on the AR background outside of the region defined by the VoyagerARDespill Mask. This is the default setting.
   - **Holdout** - to remove any green reflection in the region defined by the VoyagerARDespill Mask.
4. Then, in the **Outliner**, select the **VoyagerARDespillMask** actor.
5. In the **Output** window, move the **VoyagerARDespillMask** from its default location so that the green reflection is removed.
6. With the **VoyagerARDespillMask** still selected, in the **Details** tab, in the **Materials** section, double-click the **ARComposureFeather_Inst** material to open the editor.
7. In the editor, in the **Details** tab, in the **Feather** section, select an edge of the mask and adjust the value to blur the edge.
8. Adjust each edge as necessary to soften any hard lines.
9. Then select **Save** and close the material editor.

Virtual LED Template

With the Virtual LED template you can render content on multiple displays simultaneously. This type of project typically requires several Voyager engines. One engine is identified as the "master node" and additional engines are identified as "cluster nodes".

A Virtual LED setup outputs directly to the video wall from an HDMI or DisplayPort connection.

The following tools will help you make the most of your Virtual LED project:

- The nDisplay plugin inside Voyager communicates and synchronizes information amongst all the Voyger engines that make up the cluster, to make sure all engines render correctly at the same time (enabled by default).
- The Voyager Switchboard Launcher application (located on the master node only) is used to automatically launch and quit your Voyager project on all the engines in your setup.
- The Voyager Switchboard Listener application (running on the master node and each cluster node) listens for incoming requests from Voyager Switchboard Launcher, and processes those requests on the local engine.
- In the Voyager level blueprint, you can set up camera switching amongst multiple cameras from a number of input sources. See Adding Camera-Switching to the Level Blueprint for more information.

Virtual LED Rendering Content on Three Screens Simultaneously

Begin by creating a project from the Virtual LED template as described in Creating a Project from a Template. Then review the following topics, which are specific to a Virtual LED project:

Configuring Your Screen Setup

Using Voyager Switchboard Launcher

Using Voyager Switchboard Listener

Configuring Your Screen Setup

In your new project, you'll need to match the screens in the project to your physical screen layout. You define most aspects of the VLED system in the **VLED_Stage Editor**. Here you will define the engines that make up your network, the size and location of the screens and the viewports to render on each engine.

The default configuration is the **VLED_Stage**. This configuration can be altered to match your physical layout.

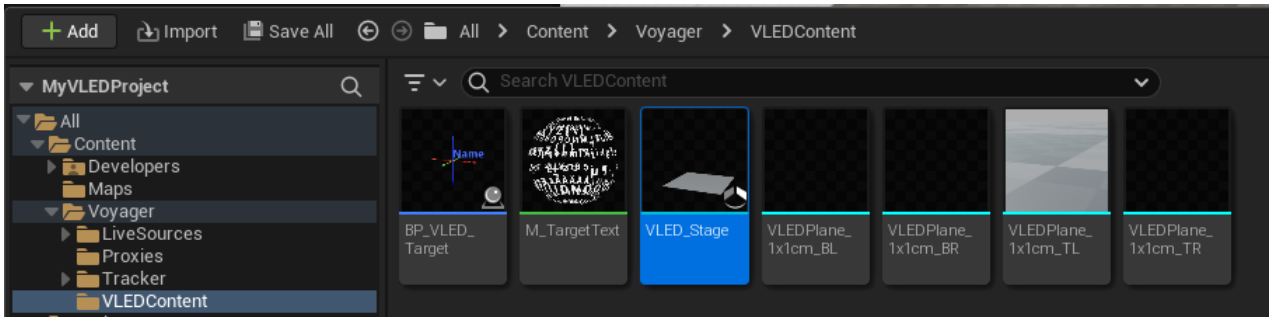
Depending on your screen setup and content, you may need more engines. In that case, you'll have to add engines to the template and assign viewports to each engine.

★When running nDisplay projects, the engine runs in **Game** mode.

★When running in **Game** mode, the project's **Media Profile** needs to be set as the **Startup Media Profile**. See Selecting the Startup Media Profile for instructions.

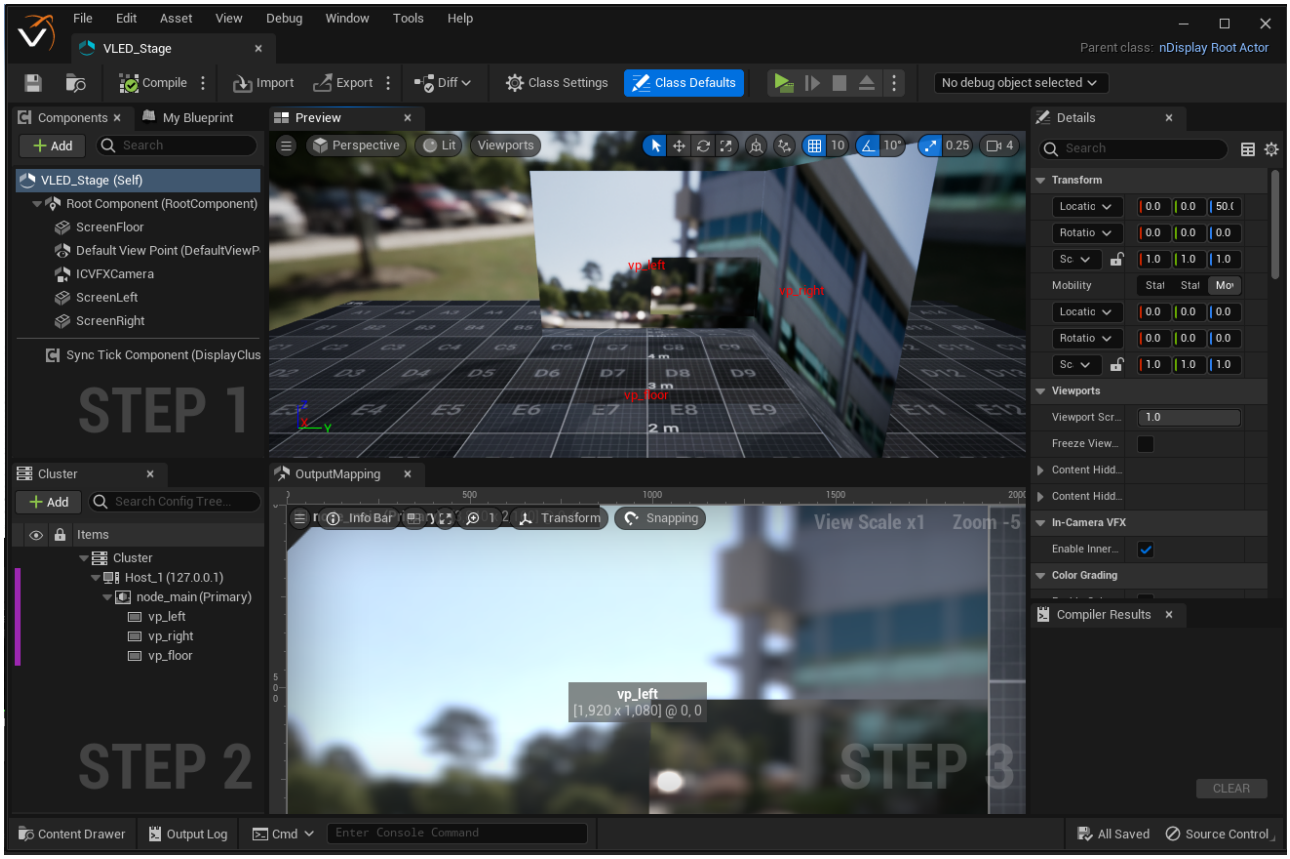
To set up the template screens:

1. In the **Content Drawer > Content > Voyager > VLEDContent**, double-click the **VLED_Stage** configuration icon to open the editor.



VLED_Stage Configuration Blueprint

The **Configuration Editor** opens.



VLED_Stage Configuration Editor

In the **Editor**, you'll see the example configuration components in the upper-left section (called **STEP 1**).

2. Define the screen parameters and add engines or screens as described in Screen Setup.

Screen Setup

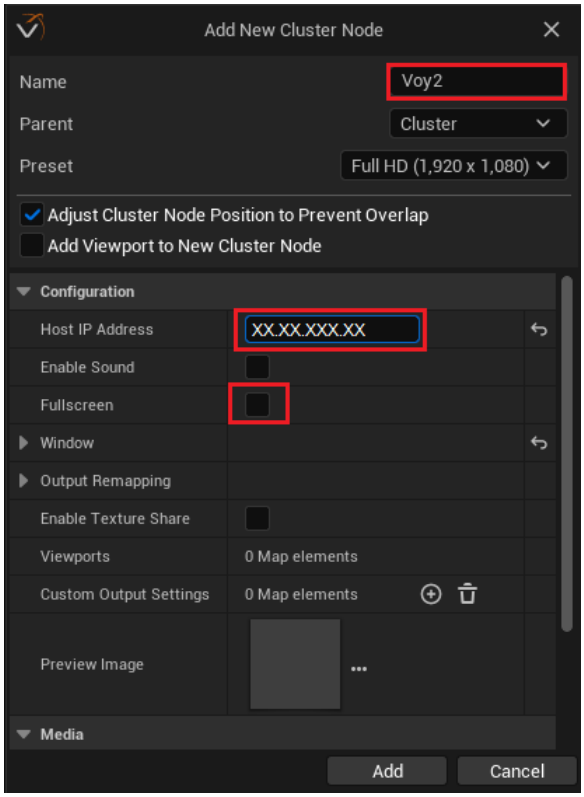
This section describes the steps for defining the screen parameters, adding new engines and screens to the project and deleting screens.

To define the screen parameters:

1. In the **VLED_Stage Configuration Editor**, in the **Preview** window or in the **Components** panel (**STEP 1**), select a screen and then move and rotate (if necessary) the screen to position it approximately where it is located in the physical screen setup.
2. In the **Details** panel, in the **Transform** section, set the **Y** and **Z Scale** values for each screen to match the size of the corresponding physical screen in your studio.
3. Measure from the zero point (**0,0**) of your studio to the bottom-left corner of each upright screen and to the front-left corner of each floor screen (if applicable) and enter these values into the appropriate **Location** fields.
4. When you have finished configuring your screens, in the main tool bar click **Save**.

To add new engines:

1. In the **Cluster** tab (**STEP 2**), select **Add New > Add New Cluster Node**.



Add New Cluster Node

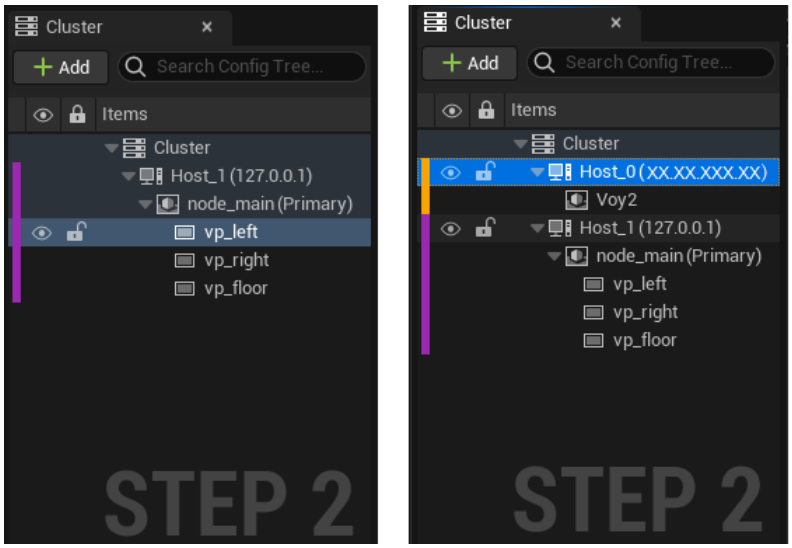
2. In the **Add New Cluster Node** editor, give the second engine a name (e.g., **Voy2**) and deselect the **Add Viewport to New Cluster Node** option.
3. In the **Configuration** section of the editor, enter the **IP Address** of the second engine.

4. Select/deselect **Fullscreen**, such that all nodes are the same, either **Fullscreen** or not **Fullscreen**.

   ★A combination of **Fullscreen** and not **Fullscreen** nodes will result in a black screen.

5. Then select **Add**.

   The new engine is added as a child of the cluster.



Add New Engine

6. To assign one or more of the viewports from the first engine to the second engine, select and drag the viewport from **Host_1 - node_main (Primary)** to the second engine.
7. Repeat the above steps to add additional engines, giving each one a unique name.

To add new screens to the blueprint:

1. In the **Components** panel (**STEP 1**), right-click any of the default screens and select **Duplicate** to make a copy.
2. With the new screen selected, in the **Details** tab on the right, rename it to correspond with one of the screens in your physical layout.
3. Move the new screen to its approximate location in the blueprint.
4. In the **Details** tab, in the **Transform** section, define the size and location of the screen as you did in the previous section for the default screens.
5. In the main tool bar select **Save**.

To delete screens:

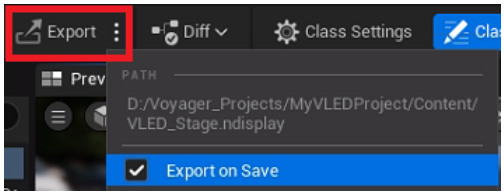
1. In the **Components** panel **(STEP 1**), right-click any of the default screens and select **Delete** to remove it.
2. In the main tool bar select **Save**.

To export the configuration file:

1. When you have finished configuring your screens, in the main toolbar of the **VLED_Stage Configuration Editor**, select **Export** to save your new configuration file (**.ndisplay**) in the **Content** folder of the project.

   The file will replace the existing **.ndisplay** file and set the export path.

2. Then select the arrow beside the **Export** icon and select **Export on Save**.



Export on Save

3. Thereafter, any time a change is saved, it will automatically update the asset and the configuration file.

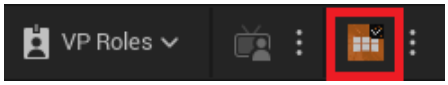
Using Voyager Switchboard Launcher

Voyager Switchboard Launcher is used to automatically launch and quit your Voyager project on all the engines in your setup simultaneously. The Voyager Switchboard Launcher is located on the master node. You will need to launch the Voyager Switchboard Listener manually on each cluster node.

To launch Voyager Switchboard Launcher:

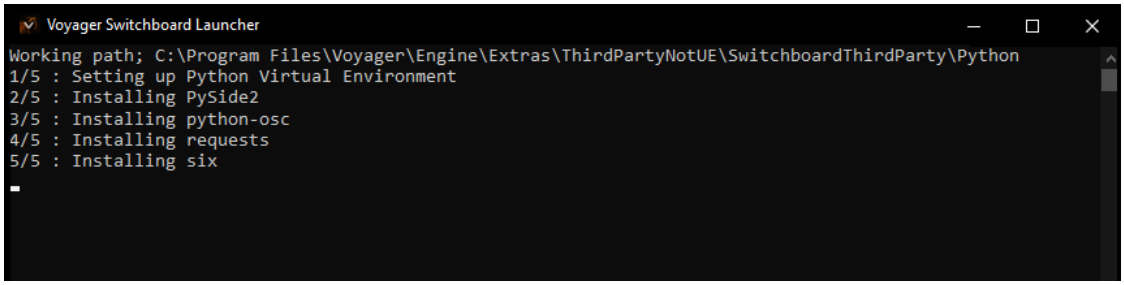
1. Select the **Voyager Switchboard Launcher** desktop icon.

   Alternatively, in the menu bar, double-click on the **Switchboard** icon.



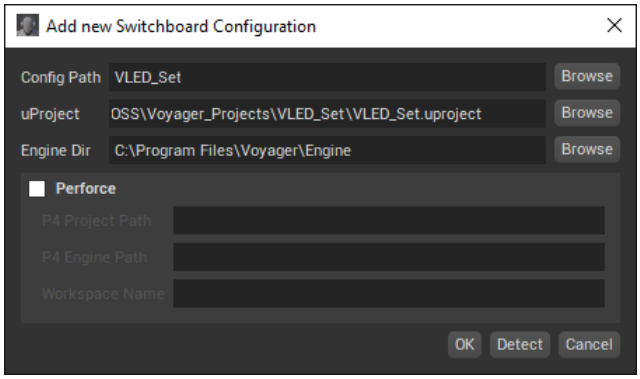
Launch Switchboard

2. When you launch **Switchboard** for the first time, it will set up a new Python virtual environment.



Python Virtual Environment Setup

   Then the **Add new Switchboard Configuration** window opens.



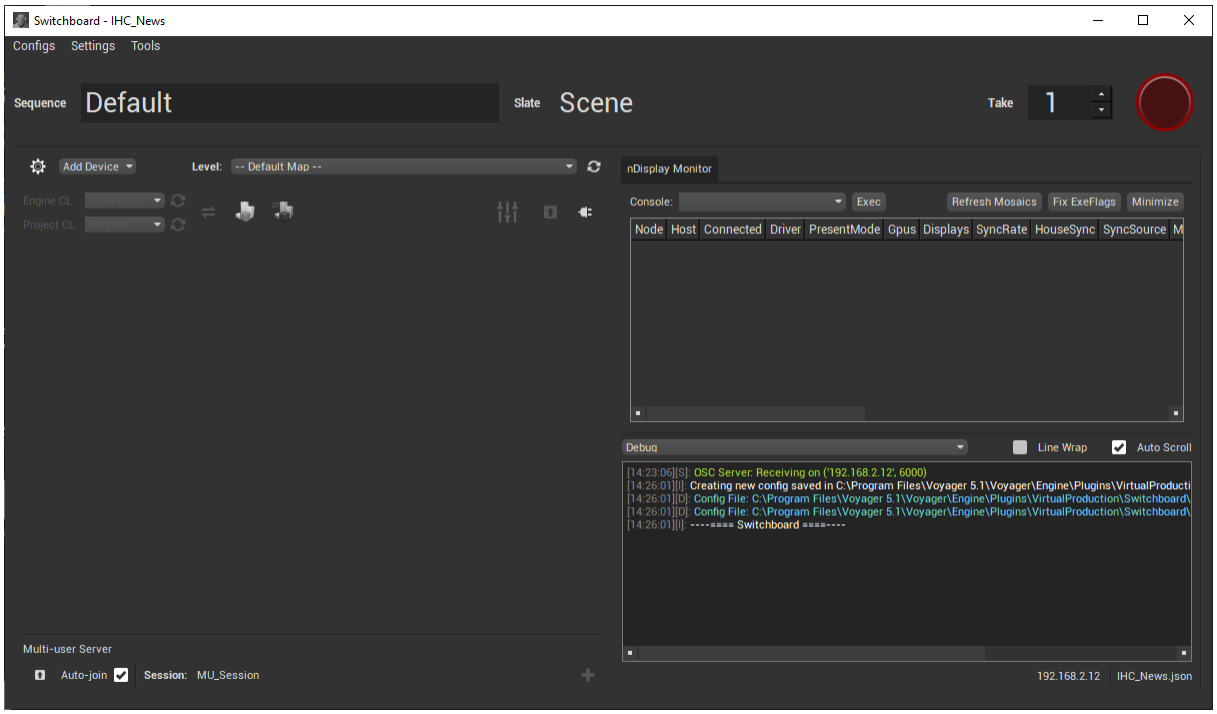
Add New Switchboard Configuration

3. The **Config Path** (the name of the configuration file), the **uProject** path and the **Engine Dir** field are automatically populated with the details of the currently open project.
4. Select **OK** to accept the automatically populated settings or edit the settings as described in the next section.

To edit the Voyager Switchboard Configuration file:

1. In the **Config Path** field, enter a new name for the **Voyager Switchboard** configuration file or select the **Browse** button and navigate to the folder containing the configuration file you want to use.
2. Beside the **uProject** field, select the **Browse** button and navigate to the project you want to control with **Switchboard**.

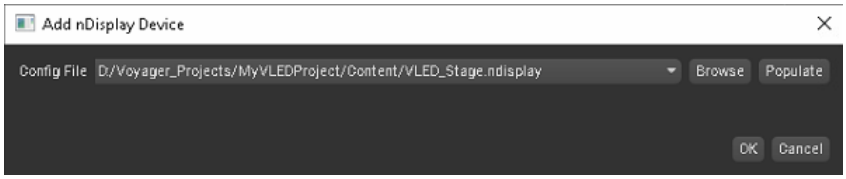
3. Beside the **Engine Dir** field, select the **Browse** button and navigate to the location of the Voyager engine on your computer.
4. Select the **Detect** button to return to the default configuration for the project that is currently open.
5. When you have finished editing the **Switchboard** configuration file, select **OK**.
6. The **Voyager Switchboard** editor opens.



Voyager Switchboard Editor

To configure Voyager Switchboard Launcher:

1. In the **Voyager Switchboard** editor, select **Add Device** and from the drop-down menu, select **nDisplay**.
2. In the **Add nDisplay Device** window, select **Browse** and navigate to the **Content** folder of your Voyager project.

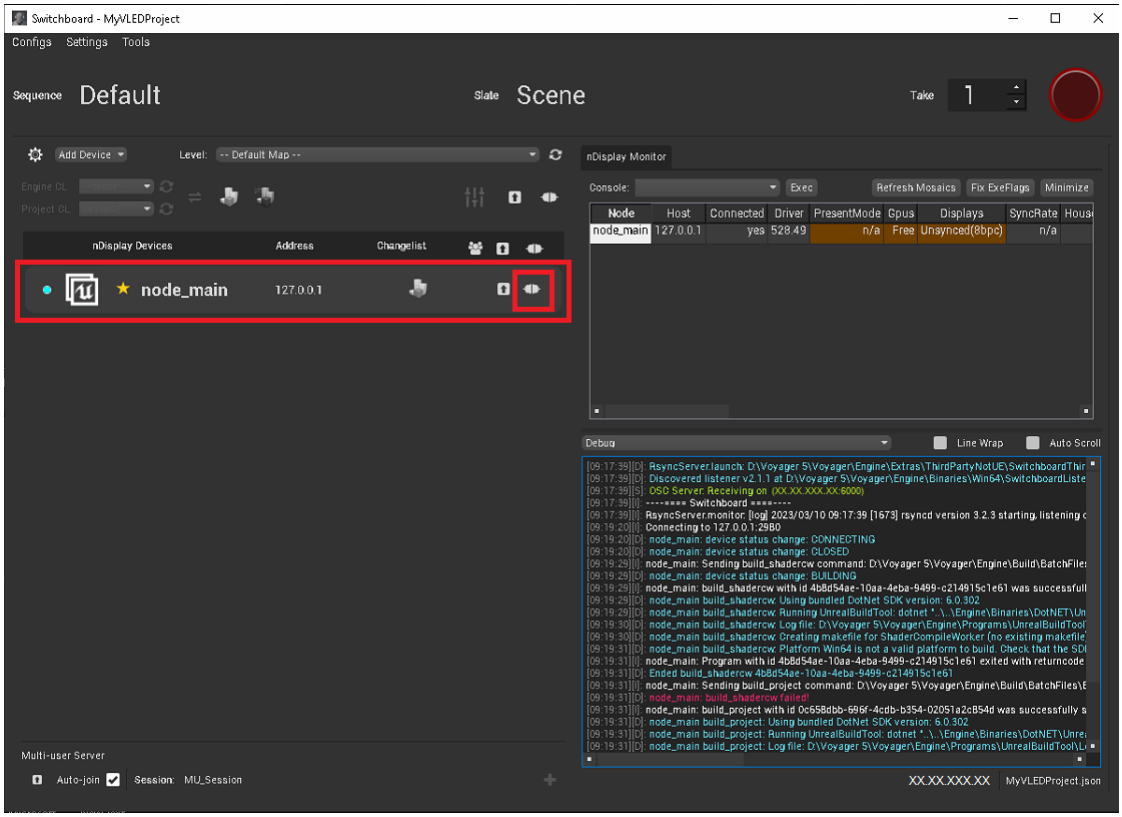


Add nDisplay Config File

3. Select the **VLEDStage.ndisplay** (or **VLED_Stage.ndisplay**) config file and select **Open**.

4. Then, in the **Add nDisplay Device** window, select **OK**.

   Now you'll see the virtual LED node displayed in the **Switchboard** editor along with any **Voyager Switchboard Listeners** that are part of your system. Each engine in your system needs to have an instance of the **Voyager Switchboard Listener** installed.

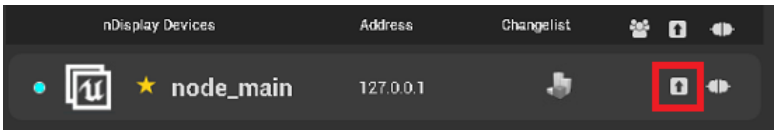


Voyager Switchboard with nDisplay Device Added

5. Select the **Connection** icon beside each component that you want to connect or select the top-level icon to connect all the components.

To start a project from Voyager Switchboard Launcher:

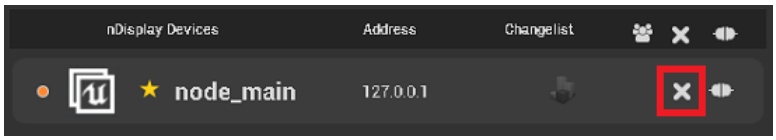
- In the list of **nDisplay Devices**, select the arrow for the device you want to start or select the top level arrow to start all connected devices.



Voyager Switchboard Launcher - Start Connected Device

To stop a project from Switchboard:

1. Press the **Windows** key on your keyboard to access the task bar.
2. Select the **Voyager Switchboard Launcher** icon to open the launcher.
3. In the list of **nDisplay Devices**, select the **X** to stop the connected device.



Voyager Switchboard Launcher - Stop Connected Device

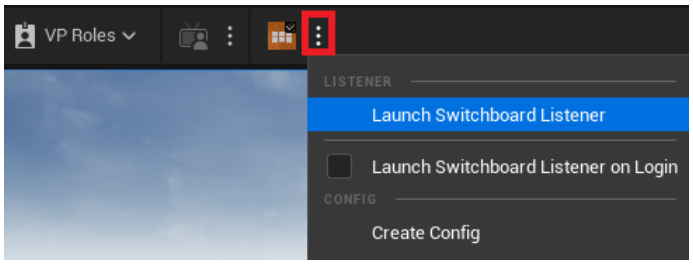
Using Voyager Switchboard Listener

While only the master node requires an instance of **Voyager Switchboard Launcher** to be running, each cluster node in your multi-display network will need to be running an instance of the **Voyager Switchboard Listener** in order to receive messages from the Launcher.

When Voyager is installed, a **Voyager Switchboard Listener** icon is automatically added to the desktop.

To launch the Voyager Switchboard Listener application:

1. Select the 3 vertical dots beside the **Launch Switchboard** icon on the desktop and select **Launch Switchboard Listener**.



Launch Switchboard Listener

2. If you get a **Windows Security Alert** message, select **Allow Access** to continue.

   The **Voyager Switchboard Listener** opens and is immediately minimized to the taskbar.

Virtual LED + Set Extension Template

With the Virtual LED template you can render content on multiple displays simultaneously and enhance your virtual set with set extension. This type of project typically requires several Voyager engines. One engine is identified as the "master node" and handles the AR or VS set extension. The additional engines are identified as "cluster nodes" and feed the LED screens.

There are three components in a Virtual LED + Set Extension setup, as follows:

The nDisplay plugin inside Voyager communicates and synchronizes information amongst all the Voyager engines that make up the cluster, to make sure all engines render correctly at the same time (enabled by default).

The Voyager Switchboard Launcher application (located on the master node only) is used to automatically launch and quit your Voyager project on all the engines in your setup.

The Voyager Switchboard Listener application (running on the master node and each cluster node) listens for incoming requests from Voyager Switchboard Launcher, and processes those requests on the local engine.

Virtual LED Rendering Content on Three Screens Simultaneously

Begin by creating a project from the **Virtual LED + Set Extension** template as described in Creating a Project from a Template. Then continue with the following steps, which are specific to a **Virtual LED + Set Extension** project:

Configuring Your Screen Setup

Capturing Shadows and Reflections

Using Voyager Switchboard Launcher

Using Voyager Switchboard Listener

Using Voyager Color Calibration

Configuring Your Screen Setup

In your new project, you'll need to match the screens in the project to your physical screen layout. You define most aspects of the system in the **VLEDStage Editor**. Here you will define the engines that make up your network, the size and location of the screens and the viewports to render on each engine.

The **VLED+Ext** template default configuration is the **VLEDStage**. This configuration can be altered to match your physical layout.

If you modify the **VLEDStage** screen setup, you'll have to also modify the **BP_VoyagerGreenScreen** actor so that it matches the screen setup in size and position.

For more information about configuring a green screen, see Using the Voyager Green Screen Model.

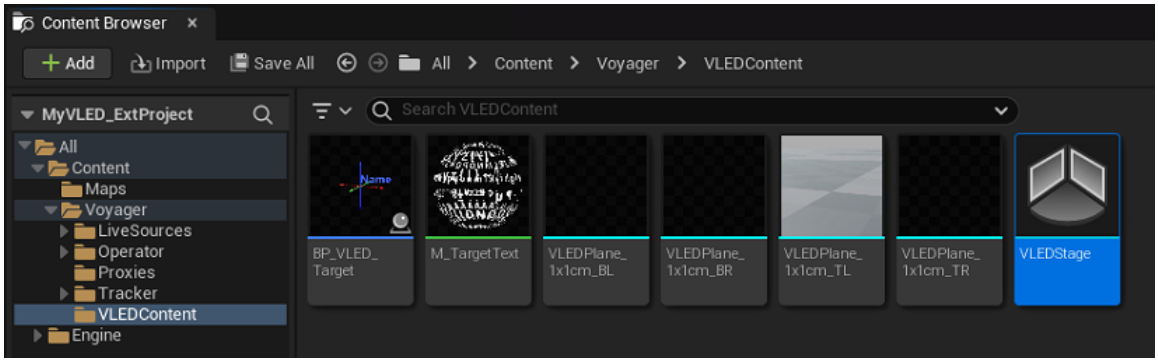
The **VLEDStage** blueprint is configured for two engines. Depending on your screen setup and content, you may need more engines. In that case, you'll have to add engines to the template and assign viewports to each engine.

★When running nDisplay projects, the engine runs in **Game** mode.

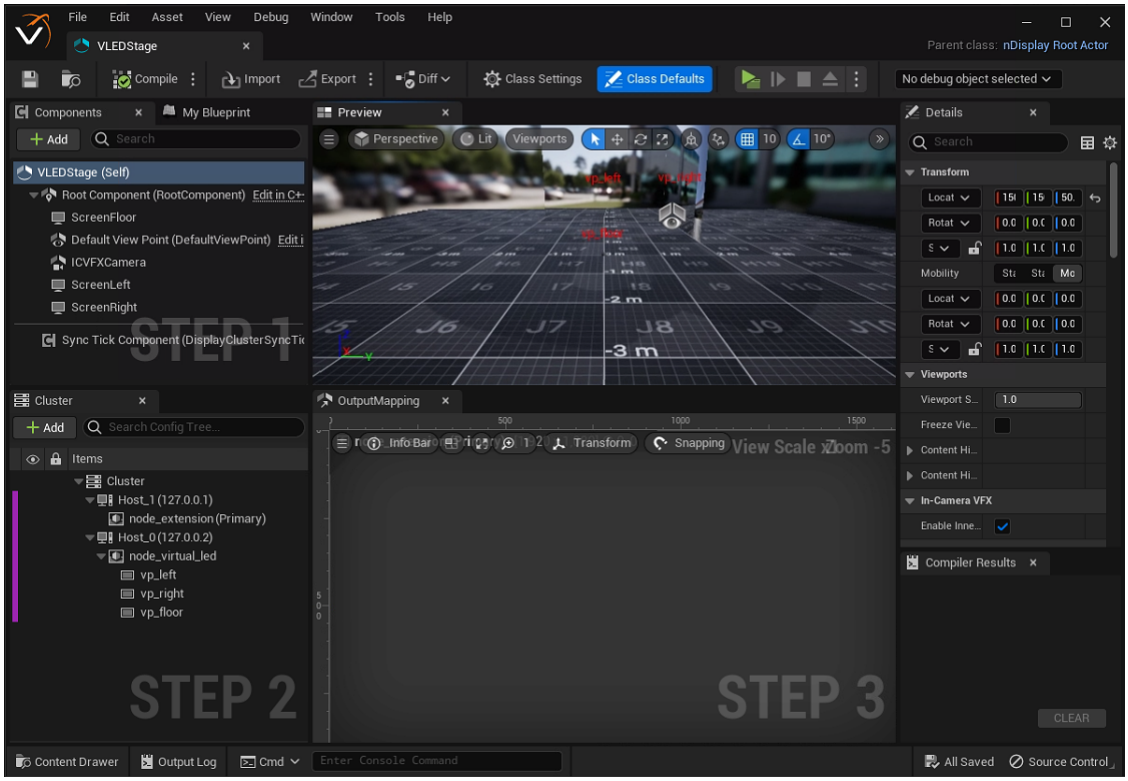
★When running in **Game** mode, the project's **Media Profile** needs to be set as the **Startup Media Profile**. See Selecting the Startup Media Profile for instructions.

To add engines:

1. In the **Content Drawer > Content > Voyager > VLEDContent**, double-click the **VLEDStage** configuration icon to open the editor.



The **Configuration Editor** opens.



VLEDStage - Configuration Editor

In the **Editor**, you'll see the example configuration components in the upper-left section (called **STEP 1**).

2. Define the screen parameters and add engines or screens as described in Screen Setup.

Screen Setup

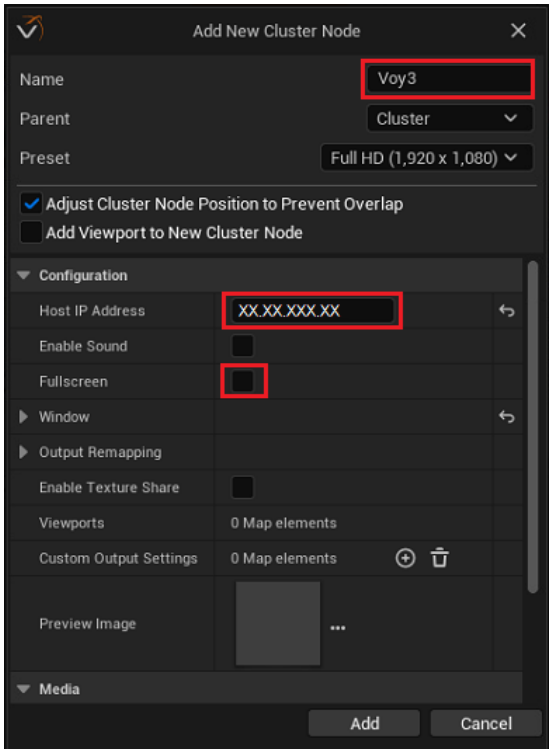
This section describes the steps for defining the screen parameters, adding new engines and screens to the project and deleting screens.

To define the screen parameters:

1. In the **VLEDStage Editor**, in the **Preview** window or in the **Components** panel (**STEP 1**), select a screen and then move and rotate (if necessary) the screen to position it approximately where it is located in the physical screen setup.
2. In the **Details** panel, in the **Transform** section, set the **Y** and **Z Scale** values for each screen to match the size of the corresponding physical screen in your studio.
3. Measure from the zero point (0,0) of your studio to the bottom-left corner of each upright screen and to the front-left corner of each floor screen (if applicable) and enter these values into the appropriate **Location** fields.
4. When you have finished configuring your screens, in the main tool bar select **Save**.

To add new engines:

1. In the **Cluster** tab (**STEP 2**), select **Add New > Add New Cluster Node**.



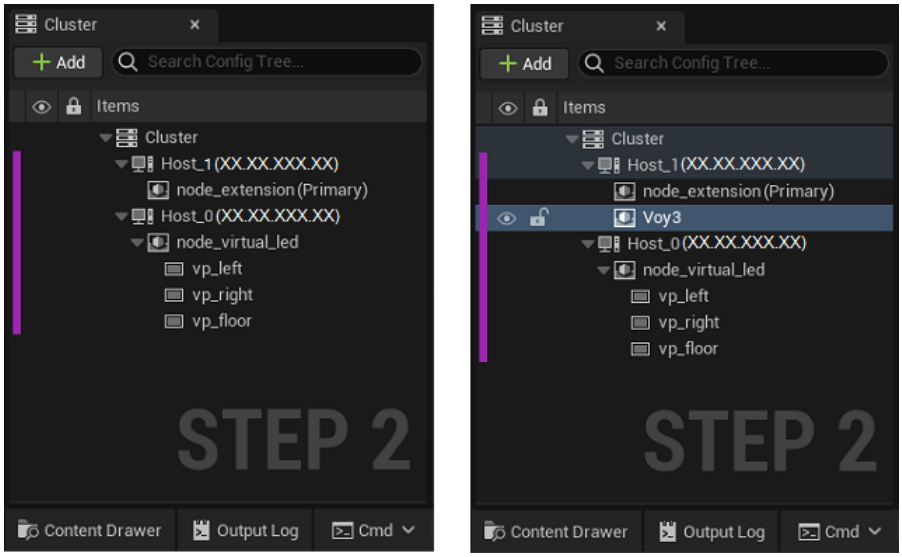
Add New Cluster Node

2. In the **Add New Cluster Node** editor, give the third engine a name (e.g., **Voy3**) and deselect the **Add Viewport to New Cluster Node** option.
3. In the **Configuration** section of the editor, enter the **IP Address** of the third engine.

4. Select/deselect **Fullscreen**, such that all nodes are the same, either **Fullscreen** or not **Fullscreen**.

   ★A combination of **Fullscreen** and not **Fullscreen** nodes will result in a black screen.

5. Then select **Add**.



Add New Engine

6. To assign one or more of the viewports from the second engine to the third engine, select and drag the viewport from **Host_0 - node_virtual_led** to the third engine.
7. Repeat the above steps to add additional engines, giving each one a unique name.

To add new screens to the blueprint:

1. In the **Components** panel (**STEP 1**), right-click any of the default screens and select **Duplicate** to make a copy.
2. With the new screen selected, in the **Details** tab on the right, rename it to correspond with one of the screens in your physical layout.
3. Move the new screen to its approximate location in the blueprint.
4. In the **Details** tab, in the **Transform** section, define the size and location of the screen as you did in the previous section for the default screens.
5. In the main tool bar select **Save**.

To delete screens:

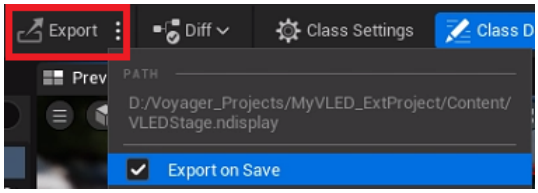
1. In the **Components** panel (**STEP 1**), right-click any of the default screens and select **Delete** to remove it.
2. In the main tool bar select **Save**.

To export the configuration file:

1. When you have finished configuring your screens, in the main toolbar of the **VLEDStage Configuration Editor**, select **Export** to save your new configuration file (**.ndisplay**) in the **Content** folder of the project.

   The file will replace the existing **.ndisplay** file and set the export path.

2. Then select the arrow beside the **Export** icon and select **Export on Save**.



Export on Save

3. Thereafter, any time a change is saved, it will automatically update the asset and the configuration file.

Capturing Shadows and Reflections

When there is a floor in your Virtual LED + Set Extension project that is part of the live feed, you will want to capture the shadows and reflections of some of the AR elements and apply them to the floor. You won't necessarily want all the virtual elements to cast shadows because part of the extension (the virtual set) is behind the talent and shouldn't cast unrealistic shadows or reflections on the talent. The rest of the virtual set does need to cast shadows and reflections on the floor.

You can use an **AR Shadow Catcher** actor for this, but you'll want to pick which AR objects will cast shadows and reflections on the **Shadow Catcher**.

Shadows

For shadows you can use **Lighting Channels**. The **Shadow Catcher** actor can use any lighting channel except channel 1 and the chosen directional lights and AR objects will use the same channel. Lighting from other lights and objects in the set that are using channel 1 won't affect the **Shadow Catcher**.

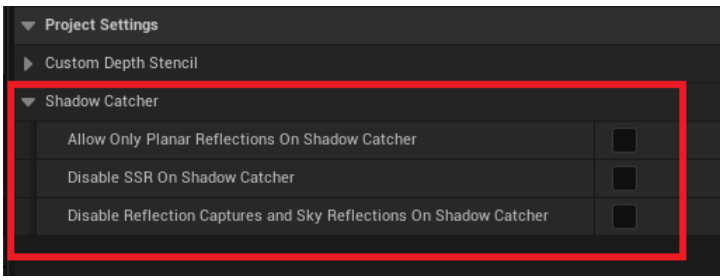
Reflections

For reflections, we can use a native **Planar Reflections** actor, place it over the **Shadow Catcher** (to get the same perspective), and use the **Planar Reflections** actor options to select which objects to capture the reflections from. However, the engine still combines the plane reflections with the global reflections (these could be Screen Space Reflections, Ray Tracing or Lumen), so that using a **Planar Reflections** actor is not enough to fully exclude the other actors from affecting the **Shadow Catcher**. In addition, there are other forms of reflection captures that could be present in the set, as well as sky reflections, fog, etc. that the user may not want to have showing on the **Shadow Catcher**.

In the **Planar Reflections** actor, use the **Use Show Only List** option, combined with a list of **Show Only Actors**.

In the Voyager Operator, the following **Shadow Catcher** options are available:

- **Allow Only Planar Reflections On Shadow Catcher** — Only the reflections from **Planar Reflections** actors will show on the Shadow Catchers (applies to all Shadow Catchers). All other types of reflections, sky reflections, sky light, fog, etc. will be prevented from showing on the Shadow Catcher. This is the recommended option to achieve reflections only for certain objects and nothing else. When this option is enabled, the next options are not available.
- **Disable SSR On Shadow Catcher** — Screen space reflections will not affect the **Shadow Catchers**, but other reflections beyond the **Planar Reflections** might, if applicable.
- **Disable Reflections Captures and Sky Reflections On Shadow Catcher** — Reflection captures and most of the reflection environment will not affect the **Shadow Catchers**. Other types of reflections, like screen space reflections, might if applicable.



Voyager Operator - Shadow Catcher Options

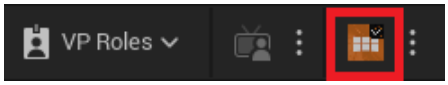
Using Voyager Switchboard Launcher

Voyager Switchboard Launcher is used to automatically launch and quit your Voyager project on all the engines in your setup simultaneously. The Voyager Switchboard Launcher is located on the master node. You will need to launch the Voyager Switchboard Listener manually on each cluster node.

To launch Voyager Switchboard Launcher:

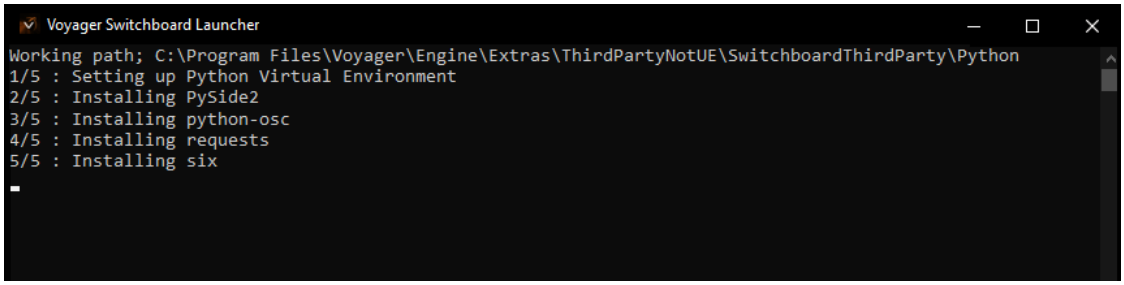
1. Select the **Voyager Switchboard Launcher** desktop icon.

   Alternatively, in the menu bar, double-click on the **Switchboard** icon.



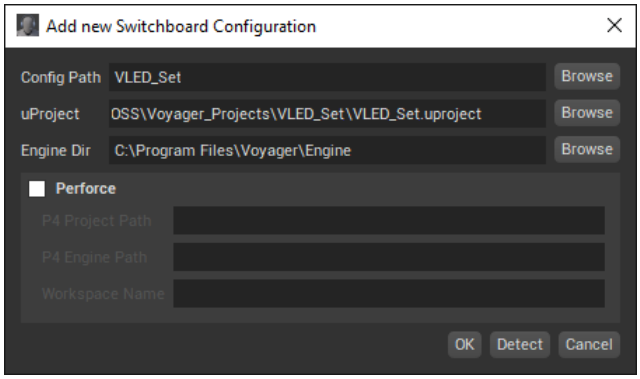
Launch Switchboard

2. When you launch **Switchboard** for the first time, it will set up a new Python virtual environment.



Python Virtual Environment Setup

   Then the **Add new Switchboard Configuration** window opens.



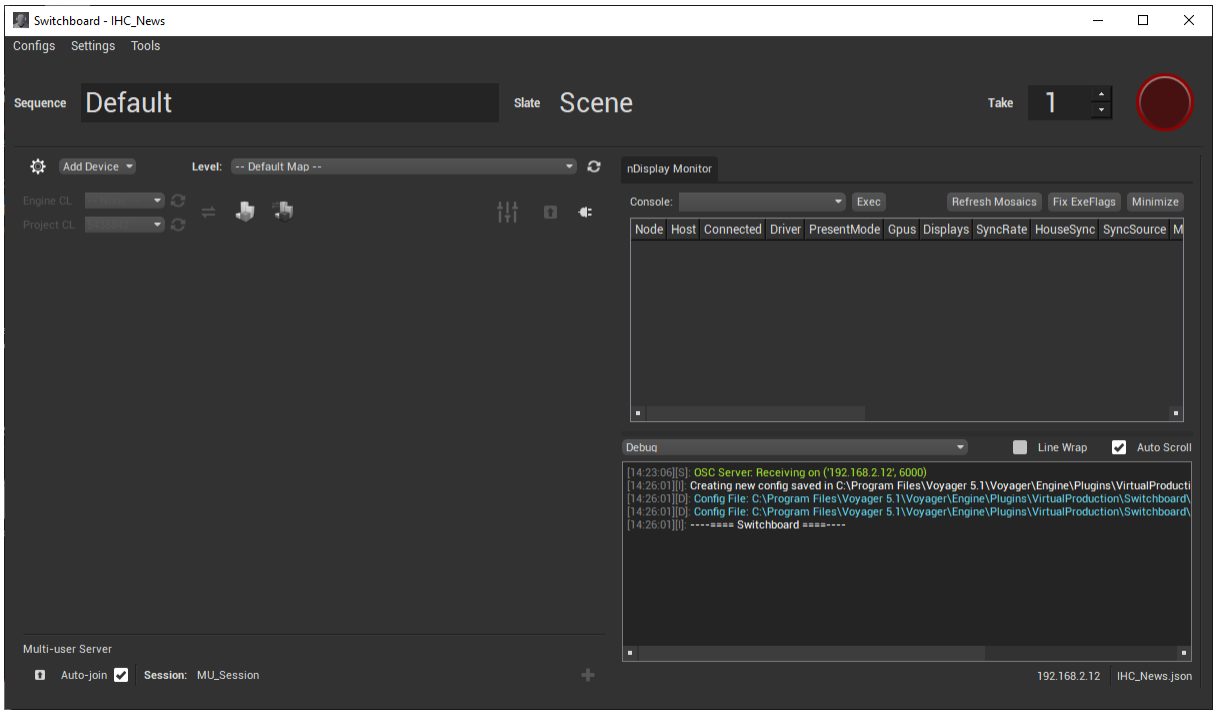
Add New Switchboard Configuration

3. The **Config Path** (the name of the configuration file), the **uProject** path and the **Engine Dir** field are automatically populated with the details of the currently open project.
4. Select **OK** to accept the automatically populated settings or edit the settings as described in the next section.

To edit the Voyager Switchboard Configuration file:

1. In the **Config Path** field, enter a new name for the **Voyager Switchboard** configuration file or select the **Browse** button and navigate to the folder containing the configuration file you want to use.
2. Beside the **uProject** field, select the **Browse** button and navigate to the project you want to control with **Switchboard**.

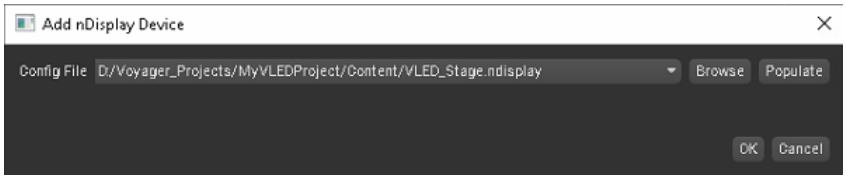
3. Beside the **Engine Dir** field, select the **Browse** button and navigate to the location of the Voyager engine on your computer.
4. Select the **Detect** button to return to the default configuration for the project that is currently open.
5. When you have finished editing the **Switchboard** configuration file, select **OK**.
6. The **Voyager Switchboard** editor opens.



Voyager Switchboard Editor

To configure Voyager Switchboard Launcher:

1. In the **Voyager Switchboard** editor, select **Add Device** and from the drop-down menu, select **nDisplay**.
2. In the **Add nDisplay Device** window, select **Browse** and navigate to the **Content** folder of your Voyager project.

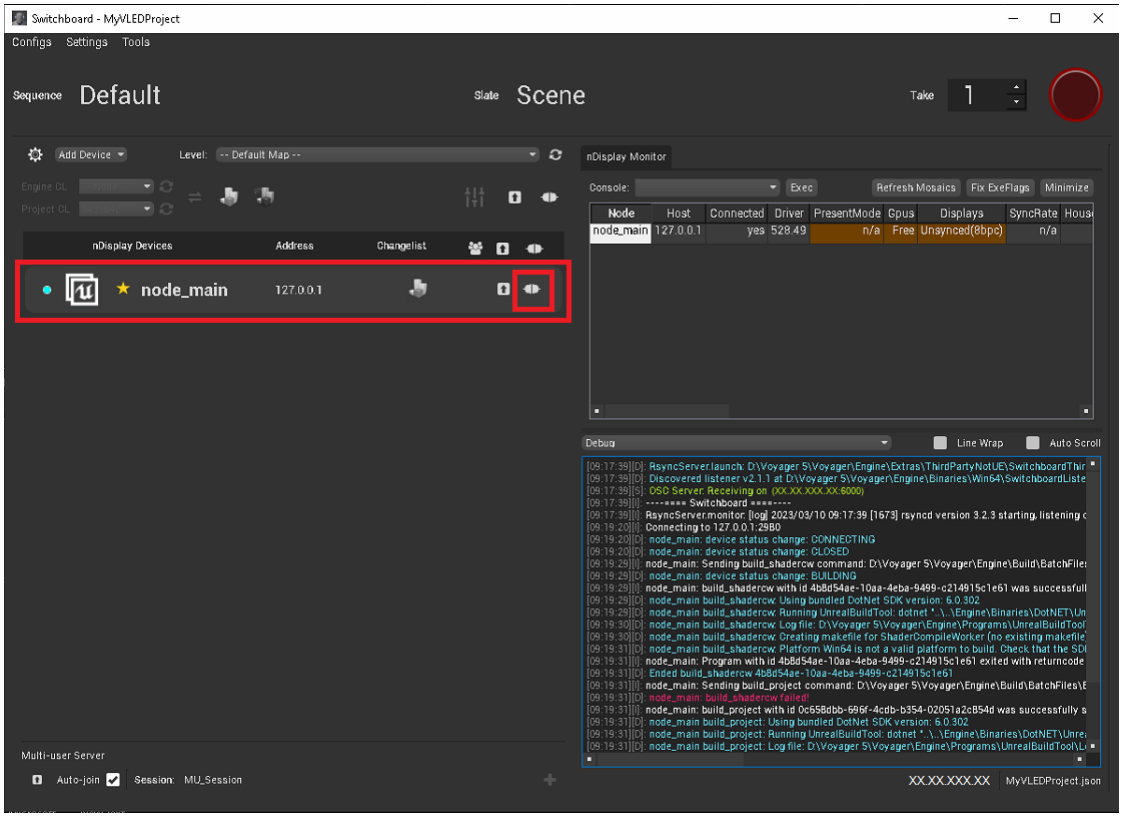


Add nDisplay Config File

3. Select the **VLEDStage.ndisplay** (or **VLED_Stage.ndisplay**) config file and select **Open**.

4. Then, in the **Add nDisplay Device** window, select **OK**.

   Now you'll see the virtual LED node displayed in the **Switchboard** editor along with any **Voyager Switchboard Listeners** that are part of your system. Each engine in your system needs to have an instance of the **Voyager Switchboard Listener** installed.

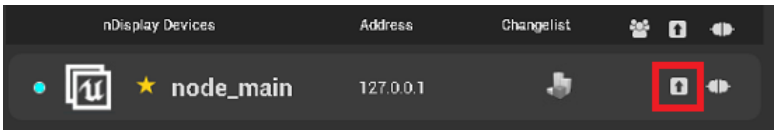


Voyager Switchboard with nDisplay Device Added

5. Select the **Connection** icon beside each component that you want to connect or select the top-level icon to connect all the components.

To start a project from Voyager Switchboard Launcher:

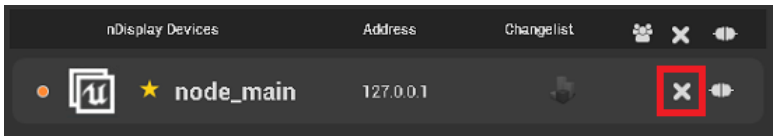
- In the list of **nDisplay Devices**, select the arrow for the device you want to start or select the top level arrow to start all connected devices.



Voyager Switchboard Launcher - Start Connected Device

To stop a project from Switchboard:

1. Press the **Windows** key on your keyboard to access the task bar.
2. Select the **Voyager Switchboard Launcher** icon to open the launcher.
3. In the list of **nDisplay Devices**, select the **X** to stop the connected device.



Voyager Switchboard Launcher - Stop Connected Device

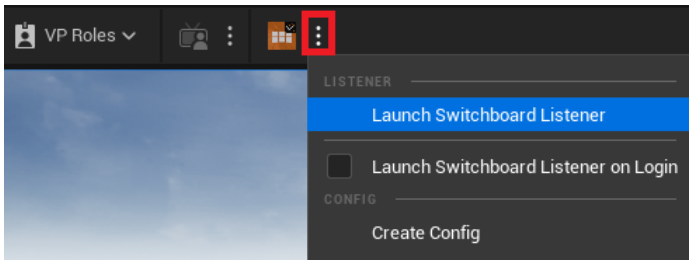
Using Voyager Switchboard Listener

While only the master node requires an instance of **Voyager Switchboard Launcher** to be running, each cluster node in your multi-display network will need to be running an instance of the **Voyager Switchboard Listener** in order to receive messages from the Launcher.

When Voyager is installed, a **Voyager Switchboard Listener** icon is automatically added to the desktop.

To launch the Voyager Switchboard Listener application:

1. Select the 3 vertical dots beside the **Launch Switchboard** icon on the desktop and select **Launch Switchboard Listener**.



Launch Switchboard Listener

2. If you get a **Windows Security Alert** message, select **Allow Access** to continue.

   The **Voyager Switchboard Listener** opens and is immediately minimized to the taskbar.

Using Voyager Color Calibration

In Voyager 5.1.1. and newer, you can apply color calibration in **VLED+Set Extension** workflows.

This process will sample multiple points in every dimension, comparing the colors in the AR part of the set with the colors in the video wall. A visual representation of the results is shown in the **Output** window.

Changes in lighting in the studio can affect the calibration, as can the LED calibration and camera settings, so if any of these things changes, the calibration would need to be redone. Once done, the calibration can be reused in other projects, as long as these factors remain the same.

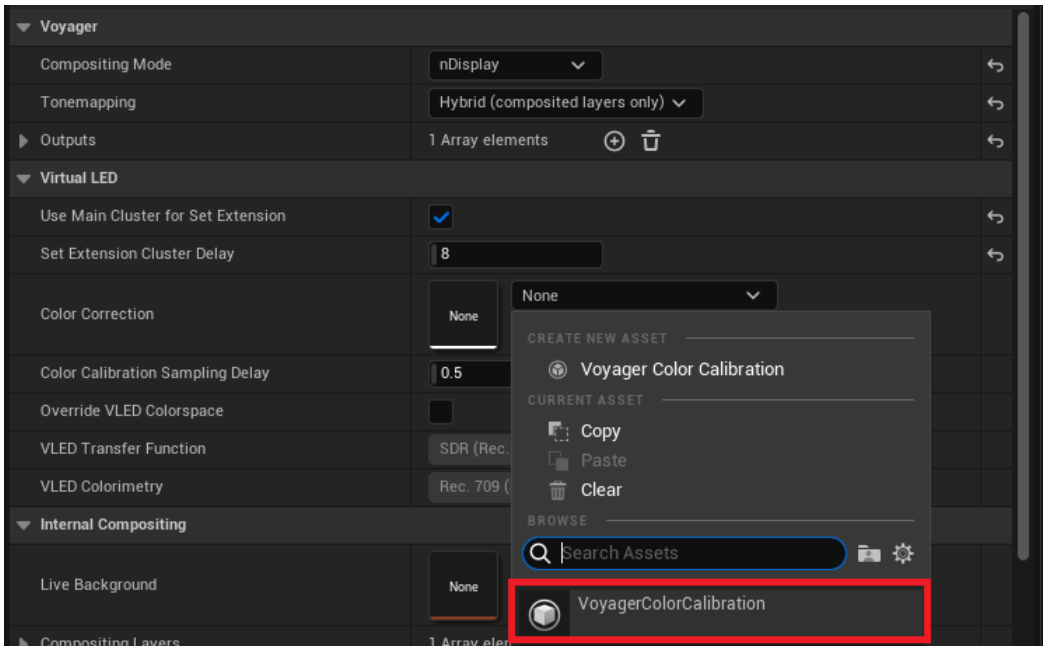
The process takes about 15 minutes.

In a good calibration, most of the mesh will be blue.

- Blue mesh represents colors that were able to be reproduced.
- Green mesh represents colors that were mapped to the closest color possible.
- Orange mesh represents colors that are outside the sampling area.

To add the Voyager Color Calibration asset:

1. In the **Outliner**, select the **VoyagerOperator** actor and in **Details > Voyager**, double-click the **VoyagerOperator** icon to open the editor.
2. In the **VoyagerOperator** editor, in the **Virtual LED** section, from the **Color Correction** drop-down, select **Voyager Color Calibration**.



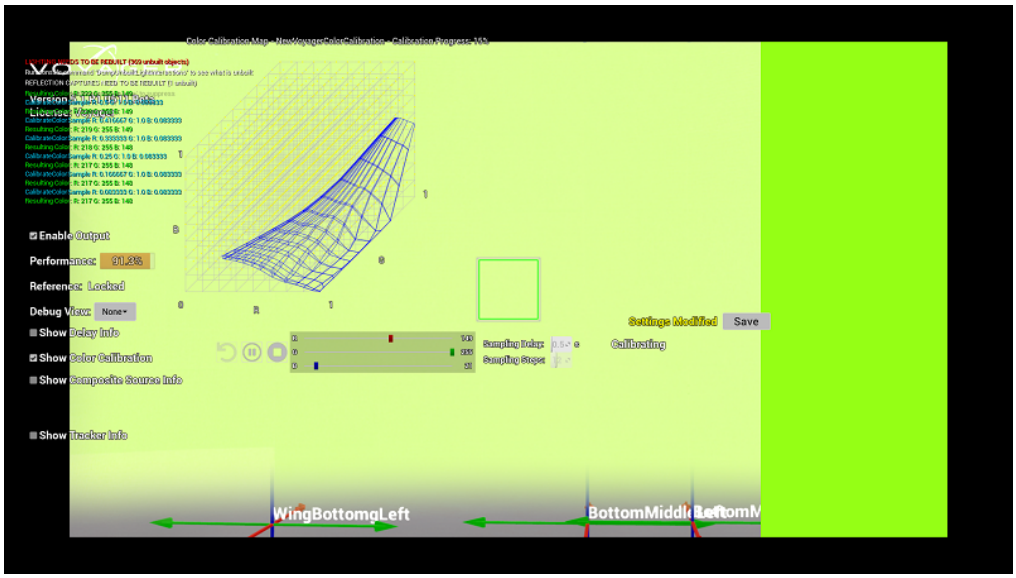
Add Voyager Color Calibration

3. Select **Save** and close the **VoyagerOperator** editor.
4. In the main Voyager screen, select **Save** again and close the project.

To run the Voyager Color Calibration:

1. Launch your project using Voyager Switchboard Launcher.
2. In the **Preview** window, select the **Show Calibration** checkbox.

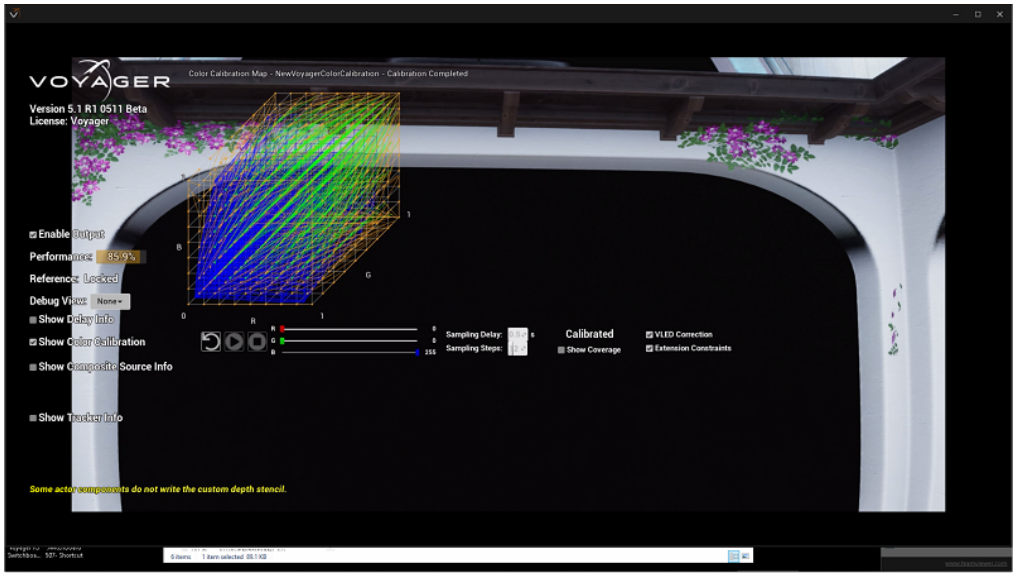
   The **Voyager Color Calibration** asset will begin sampling colors. During the calibration, the RGB sliders will show the color being calibrated, giving you an idea of the progress.



Color Sampling

3. Deselect the **Show Color Calibration** checkbox once calibration has started to save on engine performance while the system is calibrating.
4. When the calibration is finished, select the **Save** button.

   A good color calibration will show a mesh similar to the one in the image below:



Good Color Calibration

Testing the Calibration

When the calibration is completed and saved, you can use the RGB sliders to test the calibration.

To test the calibration:

1. Touch any of the **RGB** sliders to enter testing mode.
2. Move the slider to select a color to check the match between the AR extension and the LED wall.
3. Select the **Stop** button to exit testing mode.

Tips

If during the calibration process, a new color shows up on the AR side (the bar on the right) before the previous color has appeared in the sampling area (the box outline), increase the **Sampling Delay** value to give the calibration more time to sample each point and rerun the calibration.

Increasing the **Sampling Steps** value will increase the precision of the calibration but will also increase the time it takes to calibrate.

To run the calibration again:

1. Select the **Reset** button to reset and initialize the calibration and then select the **Start** button to start a new calibration.

Calibration Reset and Start

2. When the calibration is finished, select the **Save** button.

Making an Existing Project Compatible with Voyager

With this method, you'll open an existing non-Voyager project and add the necessary Voyager components.

The steps for this method are:

Setting up an Existing Project

Creating and Configuring Media Proxies

Selecting or Creating a Media Profile

Creating a Media Bundle

Creating and Configuring the Voyager Operator

Creating and Configuring the Voyager Tracker

Setting Up Compatibility With Voyager

Updating Voyager Project Settings

Organizing Voyager Assets

Setting up an Existing Project

Any Unreal Engine (UE) project can be made to work with Voyager.

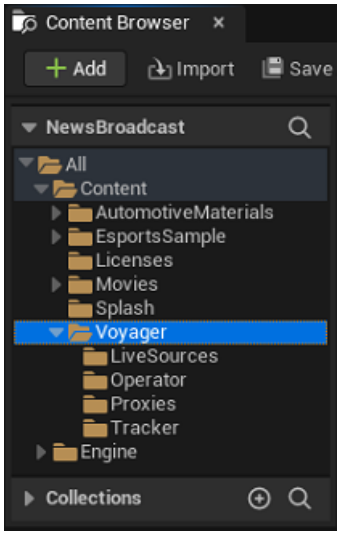
To open an existing project:

1. Navigate to the folder containing the UE project you want to use in Voyager and double-click the project file (**.uproject**).
2. In the **Select Unreal Engine Version** dialog, select the latest version and select **OK**.
3. In the **Content Browser**, if the levels aren't already displayed, select the **Filters** drop-down and from the **Filters** list, select **Level**.
4. If there is more than one level, double-click the level you want to use in your project to open it in the viewport.

To set up a Voyager folder structure for your project:

1. In the **Content Browser**, select the **Show or hide the sources panel** icon (if necessary), to display the **Content** tree.
2. Then select the **Content** folder and select the **Add New** button and from the context menu, select **New Folder**.

   Alternatively, you can right-click the **Content** folder and from the context menu, select **New Folder**.
3. Name the folder "**Voyager**".
4. With the **Voyager** folder selected, add the following sub-folders as shown in the image below:
   - LiveSources
   - Operator
   - Proxies
   - Tracker



Content Folder Tree

Creating and Configuring Media Proxies

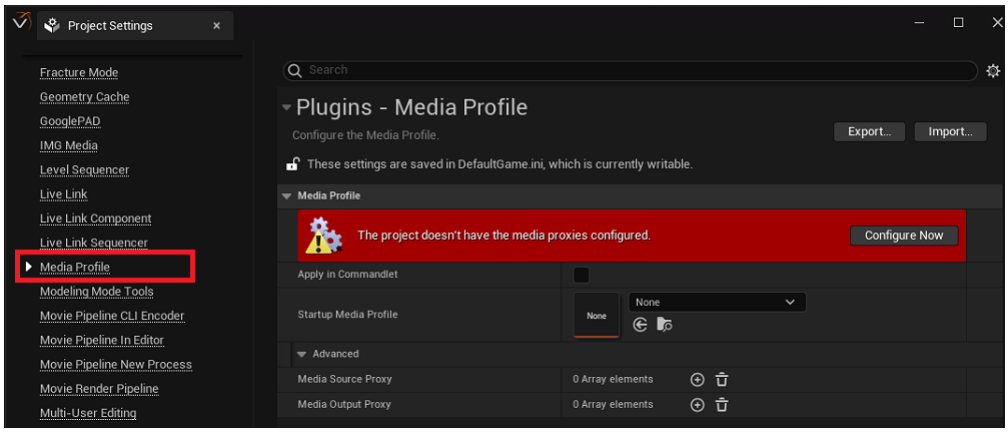
You'll need to configure media proxies to provide a connection between your media profile and the places in your project where the input(s) and output are actually used or generated.

Voyager templates include the media proxies. If you're not using a template, you need to configure media proxies for your inputs and outputs. Instructions for configuring a media proxy for a composite input and for an output are provided in this section. For instructions on configuring a media proxy for a live source, see Creating Live Sources.

To configure a media proxy for a composite input:

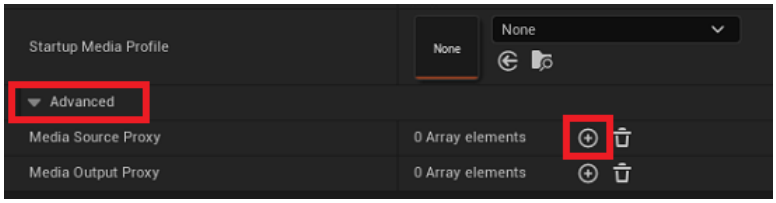
1. Select **Edit > Project Settings**, scroll down to **Plugins** and select **Media Profile**.

   The **Plugins - Media Profile** pane opens.



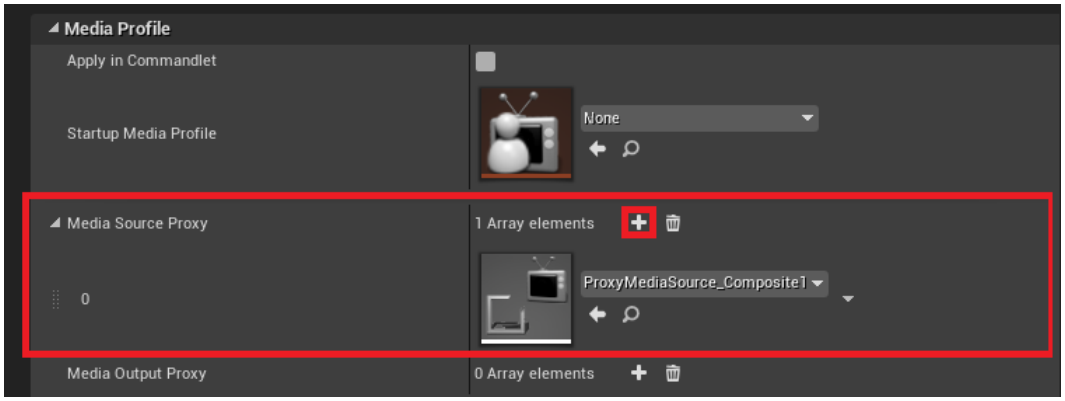
Media Profile Plugin

2. Expand the **Advanced** section and select the **+** icon on the **Media Source Proxy > Array elements** line, to add a proxy input that will correspond to the input you will create later.



Add Media Input Proxy

3. Select the arrow in the input drop-down and select **Create New Asset > Proxy Media Source**.

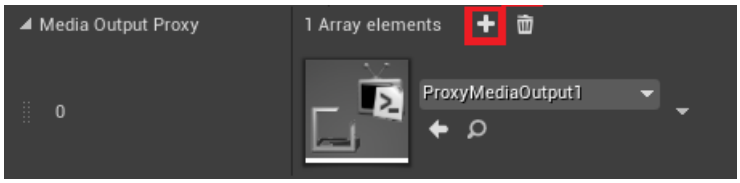


Add Proxy Media Source

4. In the **Save Asset As** window, navigate to the **Voyager > Proxies** folder and give the asset a name that will correspond to the composite input (e. g. **ProxyMediaSource_Composite1**) you will create later.
5. Select **Save**.

To configure a media proxy for an output:

1. In the **Plugins - Media Profile** pane, in the **Media Output Proxy** section, select the **+** icon to add an output proxy that will correspond to the output you will create later.



Add Media Output Proxy

2. Select the arrow in the drop-down and select **Create New Asset > Proxy Media Output**.
3. In the **Save Asset As** window, navigate to the **Voyager > Proxies** folder and give the asset a name to match the output you will create later (e. g. **ProxyMediaOutput1**).
4. Select **Save** and close the **Project Settings** editor.

Selecting or Creating a Media Profile

If you do not have a media profile on your engine, you'll need to create one. The media profile defines the number of inputs being used and configures the inputs and output for your project.

★You will not need a media profile for a Virtual LED project.

Selecting an Existing Media Profile

If you have previously created a media profile on your engine, it will be saved in **Engine Content** and is available for use with any project.

To select an existing media profile:

- Select the arrow beside the **Media Profile** icon, and select the media profile that matches your requirements.

Accessing the Media Profiles Folder

If you need to create a media profile, you'll need to access a folder in the **Engine Content**. It is best not to have the core engine files visible all the time, as inadvertently changing something in these files could interfere with your Voyager installation. Make them visible only while it is necessary, then hide them again.

To view engine content:

1. In the bottom-right corner of the **Content** space, select **View Options** and select the **Show Engine Content** checkbox.
2. Proceed with the instructions for creating a media profile.
3. When you have finished creating your media profile, go back to **View Options** and deselect the **Show Engine Content** checkbox.
4. If you want to reuse the media profile you create here in another project, you'll need to again go to **View Options**, select the **Show Engine Content** checkbox and in the **Content Browser** navigate to the profile in the **MediaProfiles** folder.

Creating a Media Profile

The steps for creating a media profile are described in the following sections:

Creating an Empty Media Profile

Configuring Inputs and Outputs

Configuring the Genlock Settings

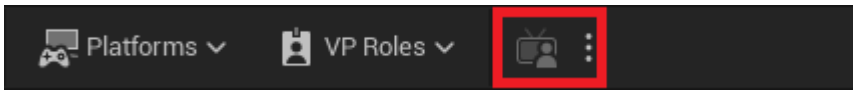
Creating an Empty Media Profile

This step creates the container that stores the input and output configurations.

You will not need a media profile for a Virtual LED project.

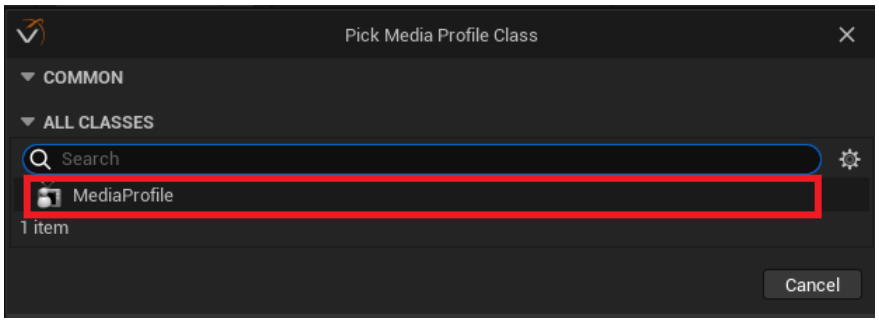
To create an empty media profile:

1. In the main toolbar, select the 3 vertical dots beside the **Media Profile** icon.



Media Profile Icon

2. From the drop-down menu, select **New Empty Media Profile**.
3. In the **Pick Media Profile Class** section, select the **MediaProfile** item and select **Select**.

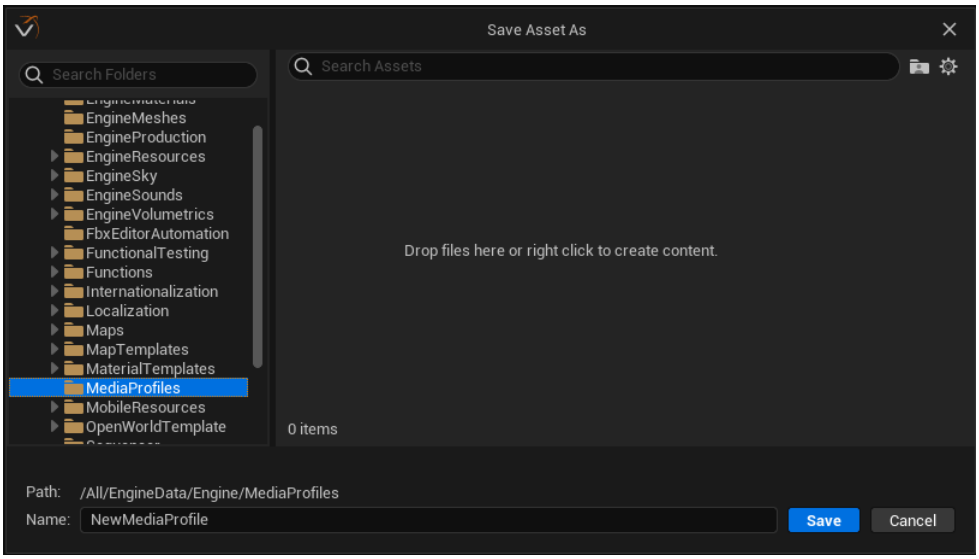


Pick Media Profile Class

4. In the **Save Asset As** window, select the settings icon to the right of the **Search Assets** field and select **Show Engine Content**.

5. Then in the **Search Folders** list, select **Content > Engine > Content**, scroll down and select the **MediaProfiles** folder.

   - If the **MediaProfiles** folder does not exist, right-click **Engine > Content** and select **New Folder** to create a folder named **MediaProfiles**.



   Save Asset As

6. In the **Name** field at the bottom of the **Save Asset As** window, enter a name for the media profile you will create and select **Save**.

   e.g., **Matrox_1080p_5994_VS**.

   The new empty media profile is saved in the **MediaProfiles** folder and the **Details** tab opens.

7. Continue with Configuring Inputs.

Configuring Inputs

Now you'll configure your input(s). The instructions in this section describe how to configure a composite input. If you want to create a live source input, see Creating Live Sources. You'll also need to create a live source material; see Creating a Live Source Material.

Select your hardware version for the appropriate instructions.

Voyager SDI and 12G

Voyager IP

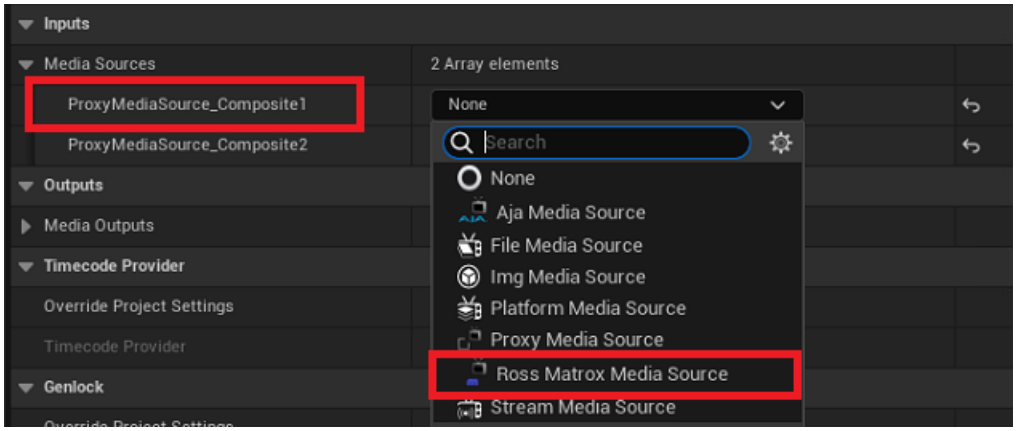
Voyager SDI and 12G

The configuration instructions in this section are for configuring a composite input for the Voyager SDI and 12G versions.

If you want to configure an input for a live source, see Creating Live Sources. You will also need to create a live source material for each live source; see Creating a Live Source Material.

To configure a composite input:

1. Double-click the **Media Profile** icon in the main tool bar to open the media profile you created, if it is not already open.
2. Expand the **Inputs > Media Sources** section, then select the **ProxyMediaSource_Composite1** drop-down and select **Ross Matrox Media Source**.



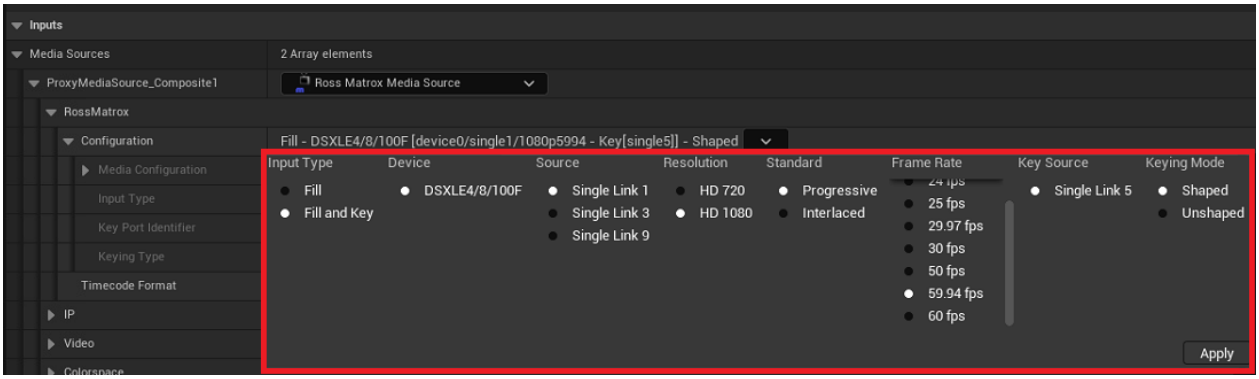
Media Source Input

3. Expand the **ProxyMediaSource_Composite1** input and then expand **RossMatrox** to access **Configuration**.

4. Select the **Configuration** drop-down and in the configuration panel that opens, select the options described below and select **Apply**.

   The **Resolution** and **Frame Rate** options will differ depending on your hardware configuration.

   - **Input Type**: Fill and Key (for a VS with an external keyer) or Fill (for VS with an internal keyer or AR)
   - **Device**: DSXLE4/8/100F
   - **Source**: Single Link 1 (first composite), Single Link 3 (2nd composite), etc.
   - **Resolution/Standard/Frame Rate**: The video format that corresponds to your workflow.
   - **Key Source**: The appropriate key source will be automatically selected, based on your selection of **Source**, if using a Matrox card. If using an AJA card, select the appropriate key source manually.
   - **Keying Mode**: Select **Shaped** (pre-multiplied) or **Unshaped** (straight alpha).



   Input Configuration - Composite Input

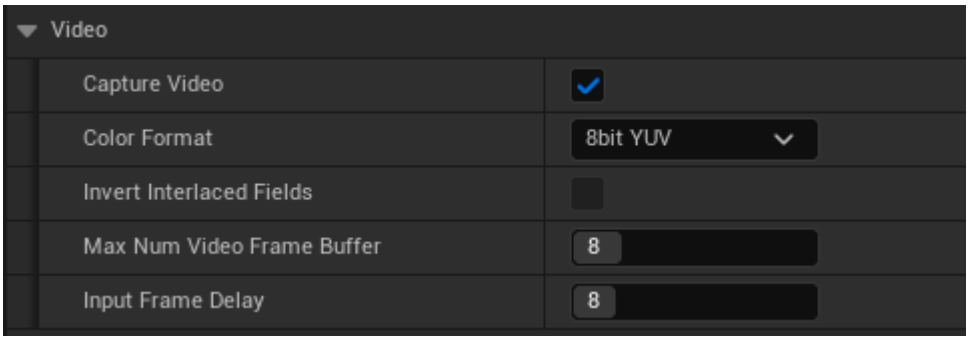
5. If you intend to use the **Timecode Provider**, from the **Timecode Format** drop-down, select **VITC**.

6. Expand **Video** and configure the settings as follows:
   - Select the **Capture Video** checkbox.
   - From the **Color Format** drop-down, select the color format that works best for your project.

     10bit YUV 422 is recommended when using **HDR** and **Wide Color Gamut**.

     With multiple inputs, there may be a performance cost when using **10bit YUV 422**. In this case, try using **8bit YUV 422**.

     10bit YUV 422 is not supported in a **Fill and Key** configuration with an AJA card. In this case you could use **RGBA**.
   - If you have an AJA card with the **Interlaced Standard**, and you notice field inversion, select the **Invert Interlaced Fields** checkbox.
   - From the **Max Num Video Frame Buffer** drop-down, select **8** if you are using a **Progressive** format or if you are using an **Interlaced** format, select **16**.
   - From the **Input Frame Delay** drop-down, select the frame delay that works best for your project.



Video Configuration - Composite Input

7. Expand **Colorspace** and configure the settings as follows:

   - From the **Colorimetry** drop-down, select one of the following options:

     ➢ **Rec. 709 (HD SDR)** for High Dynamic Range (increased levels in the range between bright and dark) and Standard Dynamic Range

     OR

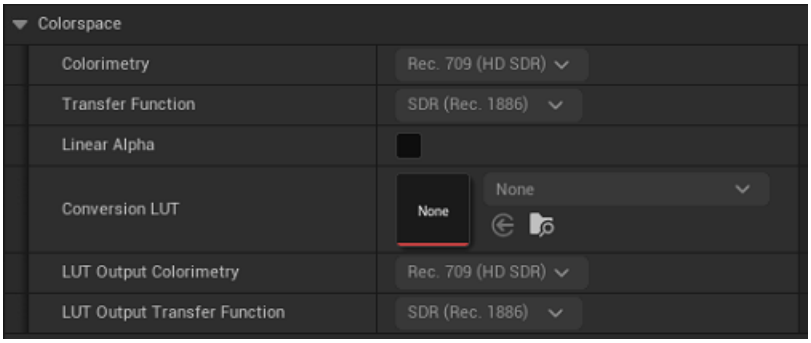
     ➢ Rec. 2020 (WCG) for Wide Color Gamut (increased selection of color values)

   - From the **Transfer Function** drop-down, select one of the following options:

     ➢ **SDR (Rec. 1886)** — Standard Dynamic Range

     ➢ **HLG (Rec. 2100)** — increases the dynamic range of the video and is compatible with both SDR and HDR displays

     ➢ **HDR10 (PQ 1000 nits)** — supports a significantly larger range of brightness as SDR, with a corresponding increase in contrast and a color palette of one billion shades.



Colorspace Settings

   - Select the **Linear Alpha** checkbox if the incoming alpha is already linear; the **Transfer Function** will not be applied.

     Refer to the documentation for your chroma keyer or key source to determine whether or not the alpha is linear.

   - If you selected an 8bit color format in the **Video** settings, these are the only **Colorspace** settings available. Select **Save** and continue with the output configuration.

   - If you selected a 10bit color format in the **Video** settings, the **HDR** settings will also be available. Continue with the next steps to edit **HDR** settings.

     ➢ From the **Conversion LUT** drop-down, browse to and select the **Look Up Table** you want to apply to your color grading.

     ➢ From the **LUT Output Colorimetry** drop-down, select either **Rec. 709 (HD SDR)** or **Rec. 2020 (WCG)**.

     ➢ From the **LUT Output Transfer Function** drop-down, select either **SDR (Rec. 1886)**, **HLG (Rec. 2100)** or **HDR10 (PQ 1000 nits)**.

8. Select **Save**.

Continue with Configuring an Output.

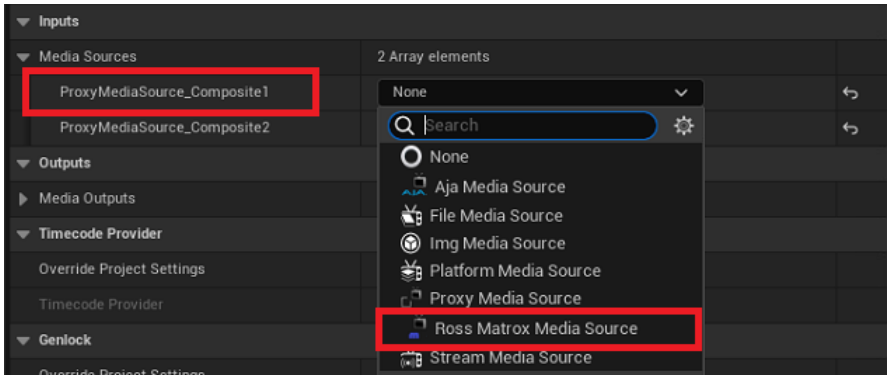
Voyager IP

The configuration instructions in this section are for the Voyager IP version.

If you want to configure an input for a live source, see Creating Live Sources. You will also need to create a live source material for each live source; see Creating a Live Source Material.

To configure a composite input:

1. Double-click the **Media Profile** icon in the main toolbar to open the media profile you created.
2. Expand the **Inputs > Media Sources** section, then select the **ProxyMediaSource_Composite1** drop-down and select **Ross Matrox Media Source**.

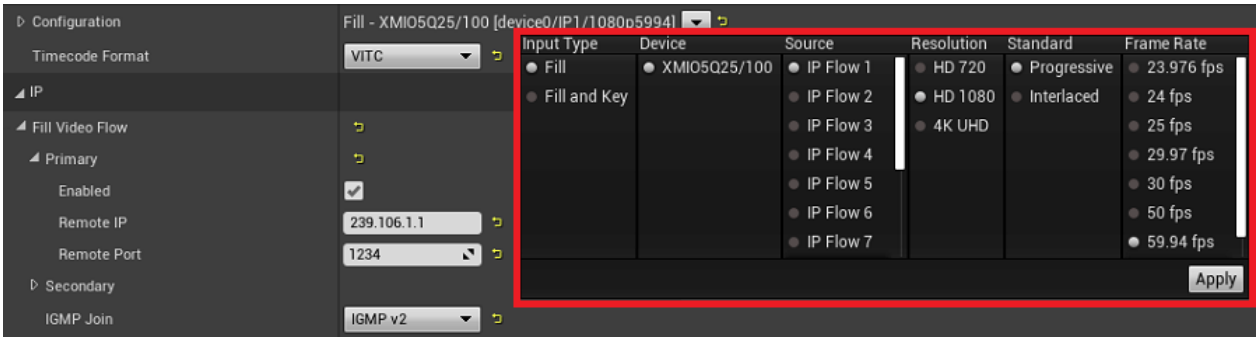


Media Source Input

3. Expand the **ProxyMediaSource_Composite1** input and then expand **RossMatrox** to access **Configuration**.
4. If you want to use the **Timecode** functionality, from the **Timecode Format** drop-down, select **VITC**.
5. Select the **Configuration** drop-down and in the configuration panel that opens, select the options described below and select **Apply**.

   The **Resolution** and **Frame Rate** options will differ depending on your hardware configuration.

   - **Input Type**: Fill and Key (for a VS with an external keyer) or Fill (for VS with an internal keyer or AR)
   - **Device**: Matrox
   - **Source**: IP Flow 1
   - **Resolution/Standard/Frame Rate**: The video format that corresponds to your workflow.



Input Configuration - Composite Input (IP)

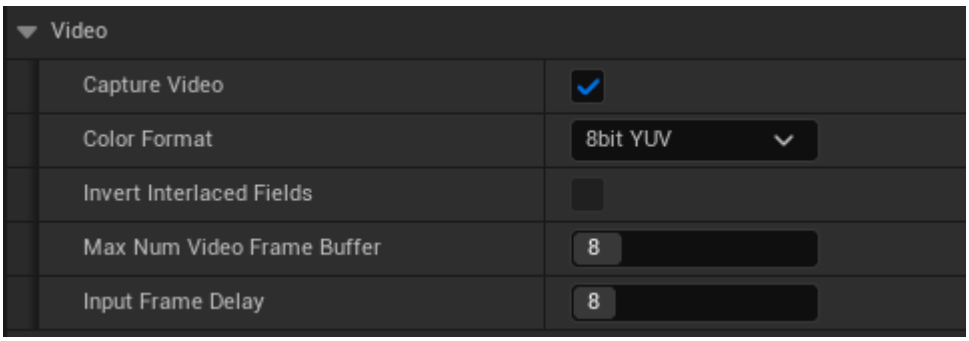
6. Expand the **IP** section and then expand **Primary** and configure the settings as follows:
   - Select the **Enabled** checkbox.
   - In the **Remote IP** field, enter the IP address of the remote machine.
   - In the Port field, enter the port number on which the machines will be communicating.
   - Expand **Secondary** and from the **IGMP Join** drop-down, select **IGMP v2**.
7. If you are using a **Matrox IP 2110** card and require timecode functionality, expand **Ancillary Flow** and **Primary** and configure the settings as follows:
   - Select the **Enabled** checkbox.
   - In the **Remote IP** field, enter the IP address of the remote machine.
   - In the **Port** field, enter the port number on which the machines will be communicating.
   - Expand **Secondary** and from the **IGMP Join** drop-down, select IGMP v2.
8. Expand **Video** and configure the settings as follows:
   - Select the **Capture Video** checkbox.
   - From the **Color Format** drop-down, select the color format that works best for your project.

     10bit YUV 422 is recommended when using **HDR** and **Wide Color Gamut**.

     With multiple inputs, there may be a performance cost when using **10bit YUV 422**. In this case, try using **8bit YUV 422**.

     10bit YUV 422 is not supported in a Fill and Key configuration with an AJA card. In this case you could use **RGBA**.

     Leave the remaining settings as they are.
   - If you have an AJA card with the **Interlaced Standard**, and you notice field inversion, select the **Invert Interlaced Fields** checkbox.
   - From the **Max Num Video Frame Buffer** drop-down, select **8** if you are using a **Progressive** format or if you are using an Interlaced format, select **16**.
   - From the **Input Frame Delay** drop-down, select the frame delay that works best for your project.



Video Configuration - Composite Input

9. Expand **Colorspace** and configure the settings as follows:

   - From the **Colorimetry** drop-down, select one of the following options:

     ➢ **Rec. 709 (HD SDR)** for High Dynamic Range (increased levels in the range between bright and dark) and Standard Dynamic Range

     OR

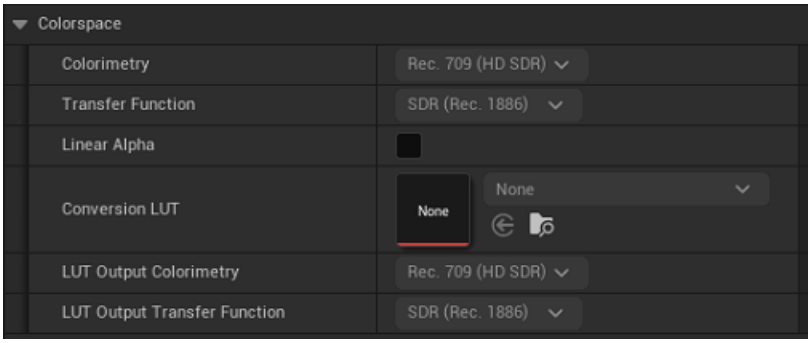
     ➢ Rec. 2020 (WCG) for Wide Color Gamut (increased selection of color values)

   - From the **Transfer Function** drop-down, select one of the following options:

     ➢ **SDR (Rec. 1886)** — Standard Dynamic Range

     ➢ **HLG (Rec. 2100)** — increases the dynamic range of the video and is compatible with both SDR and HDR displays

     ➢ **HDR10 (PQ 1000 nits)** — supports a significantly larger range of brightness as SDR, with a corresponding increase in contrast and a color palette of one billion shades.



Colorspace Settings

   - Select the **Linear Alpha** checkbox if the incoming alpha is already linear; the **Transfer Function** will not be applied.

     Refer to the documentation for your chroma keyer or key source to determine whether or not the alpha is linear.

   - If you selected an 8bit color format in the **Video** settings, these are the only **Colorspace** settings available. Select **Save** and continue with the output configuration.

   - If you selected a 10bit color format in the **Video** settings, the **HDR** settings will also be available. Continue with the next steps to edit **HDR** settings.

     ➢ From the **Conversion LUT** drop-down, browse to and select the **Look Up Table** you want to apply to your color grading.

     ➢ From the **LUT Output Colorimetry** drop-down, select either **Rec. 709 (HD SDR)** or **Rec. 2020 (WCG)**.

     ➢ From the **LUT Output Transfer Function** drop-down, select either **SDR (Rec. 1886)**, **HLG (Rec. 2100)** or **HDR10 (PQ 1000 nits)**.

10. Select **Save**.

    Continue with Configuring an Output.

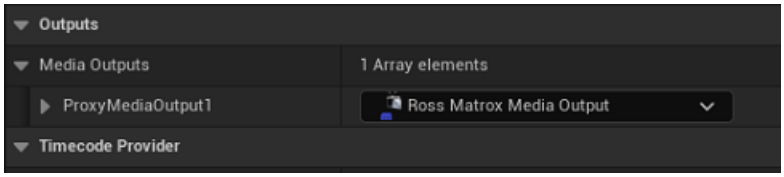
Configuring an Output

There is only one output available in Voyager. You only need to decide whether you want to configure that output for internal compositing or external compositing. Instructions for both are provided in this section.

Internal Compositing

To configure an output for internal compositing:

1. In the **Outputs** section, expand **Media Outputs**.
2. Then select the **ProxyMediaOutput1** drop-down and select **Ross Matrox Media Output**.

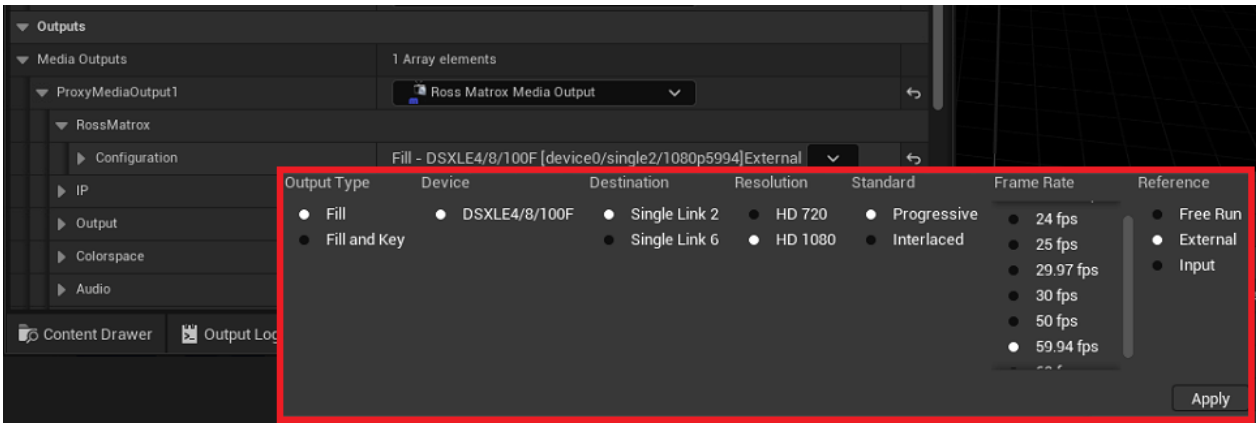


Media Profile - Output Config

3. Expand **ProxyMediaOutput1** and then expand **RossMatrox** to access **Configuration**.
4. Select the **Configuration** drop-down and in the configuration panel that opens, select the options described below and select **Apply**.

   The **Source**, **Resolution** and **Frame Rate** options will differ depending on your hardware configuration.

   - **Output Type**: Fill
   - **Device**: DSXLE4/8/100F
   - **Destination**: The physical pin you are using for your output (different cards will display different options here)
   - **Resolution/Standard/Frame Rate**: The video format that corresponds to your workflow.
   - **Reference**: External



Output Configuration

5. Expand **Output**, select the arrow in the **Pixel Format** field and select the format that works best for your project.

   10bit YUV 422 is recommended when using **HDR** and **Wide Color Gamut**.

   With multiple inputs, there may be a performance cost when using **10bit YUV 422**. In this case, try using **8bit YUV 422**.

   10bit YUV 422 is not supported in a **Fill** and **Key** configuration with an AJA card. In this case you could use **RGBA**.

6. Select the **Parallelize Transfers** checkbox to improve performance.

   Output frames are transferred faster, which can make a noticeable difference in 4K projects.

7. Leave the remaining settings as they are and select **Save**.

   Continue with Configuring the Genlock Settings.

External Compositing

To configure an output for external compositing:

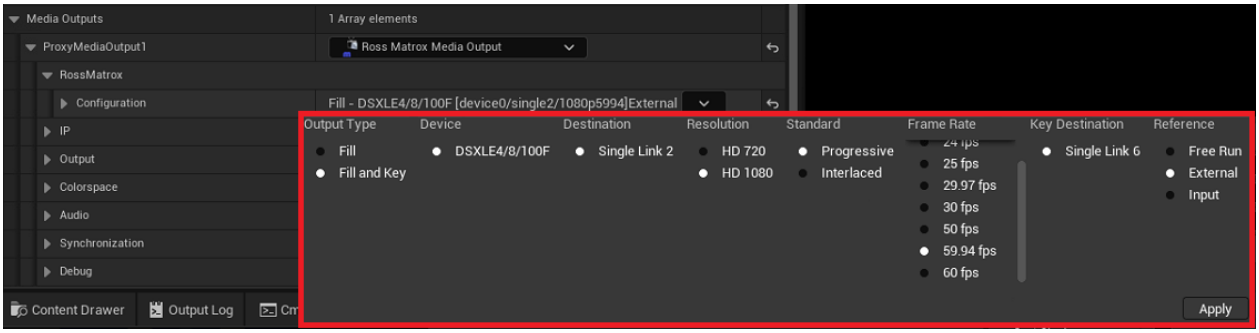
1. In the **Outputs** section, expand **Media Outputs**.
2. Then select the **ProxyMediaOutput1** drop-down and select **Ross Matrox Media Output**.
3. Expand **ProxyMediaOutput1** and then expand **RossMatrox** to access **Configuration**.
4. Select the **Configuration** drop-down and in the configuration panel that opens, select the options described below and select **Apply**.

   The **Resolution** and **Frame Rate** options will differ depending on your hardware configuration.

   - **Output Type**: Fill and Key
   - **Device**: DSXLE4/8/100F
   - **Destination**: The physical pin you are using for your output (different cards will display different options here)

     ★For AJA cards, the **Destination** and **Key Destination** pins need to be adjacent (e.g., **Destination** = Out 1 and **Key Destination** = Out 2).
   - **Resolution/Standard/Frame Rate**: The video format that corresponds to your workflow.
   - **Key Destination**: The appropriate key destination will be automatically selected, based on your selection of **Destination**, if using a Matrox card. If using an AJA card, select the appropriate key destination manually.

     ★For AJA cards, the **Destination** and **Key Destination** pins need to be adjacent (e.g., **Destination** = Out 1 and **Key Destination** = Out 2).
   - **Reference**: External

   Output Configuration - External Compositing
5. Expand **Output**, select the arrow in the **Pixel Format** field and select the format that works best for your project.

   10bit YUV 422 is recommended when using **HDR** and **Wide Color Gamut**.

   With multiple inputs, there may be a performance cost when using **10bit YUV 422**. In this case, try using **8bit YUV 422**.

   10bit YUV 422 is not supported in a **Fill and Key** configuration with an AJA card. In this case you could use **RGBA**.
6. Select the **Parallelize Transfers** checkbox to improve performance.

   Output frames are transferred faster, which can make a noticeable difference in 4K projects.

7. Leave the remaining settings as they are and select **Save**.

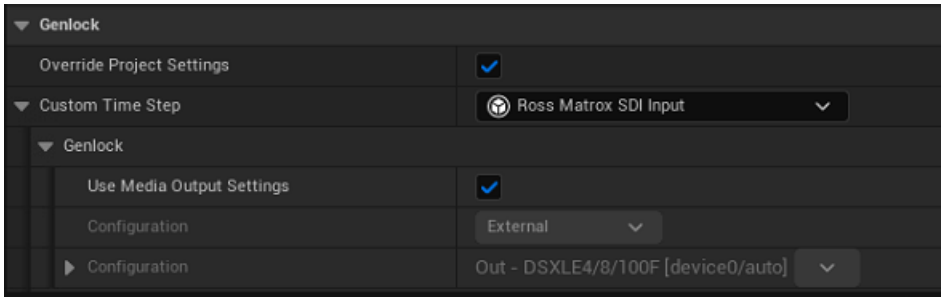
   Continue with Configuring the Genlock Settings.

Configuring the Genlock Settings

Once you have your input(s) and output configured, the last step is to configure the **Genlock** settings. The settings will depend on the type of graphics card you have in your engine. Select the appropriate instructions below:

To configure the Genlock Settings for a Matrox card:

1. In the **Genlock** section, select the **Override Project Settings** checkbox.

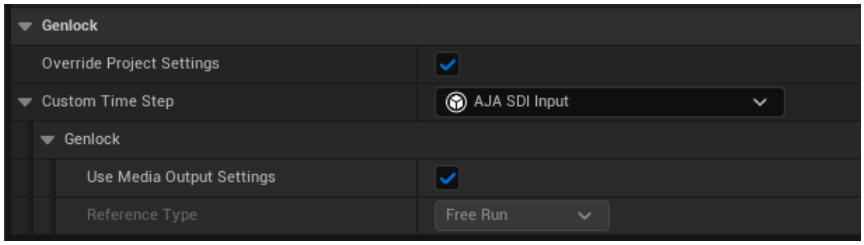


Genlock Configuration - Matrox

2. Select the **Custom Time Step** arrow and from the drop-down select **Ross Matrox SDI Input**.
3. Expand **Custom Time Step** and then expand **Genlock** and select the **Use Media Output Settings** checkbox.
4. Select **Save** and close the **Media Profile** window.

To configure the Genlock Settings for an AJA card:

1. In the **Genlock** section, select the **Override Project Settings** checkbox.



Genlock Configuration - AJA

2. Select the **Custom Time Step** arrow and from the drop-down select **AJA SDI Input**.
3. Expand **Custom Time Step** and then expand **Genlock** and select the **Use Media Output Settings** checkbox.

   This forces the AJA **Custom Time Step** to use the **Genlock** settings configured on the first **AJA Media Output** of the **Media Profile**.
4. Select **Save** and close the **Media Profile** window.

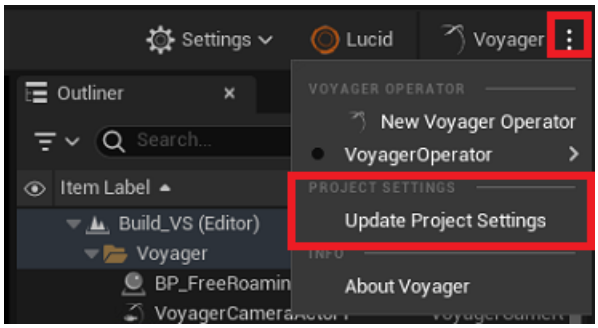
Continue with Updating Voyager Project Settings.

Updating Voyager Project Settings

Once you've set up your project, it's time to update the Voyager project settings.

To update the project settings:

1. Select the 3 vertical dots beside the Voyager icon and select **Select a Voyager Operator > VoyagerOperator**.
2. Then select **Update Project Settings**.



Updating Voyager Project Settings

3. In the **Voyager Project Defaults** dialog, select **Yes** to proceed.

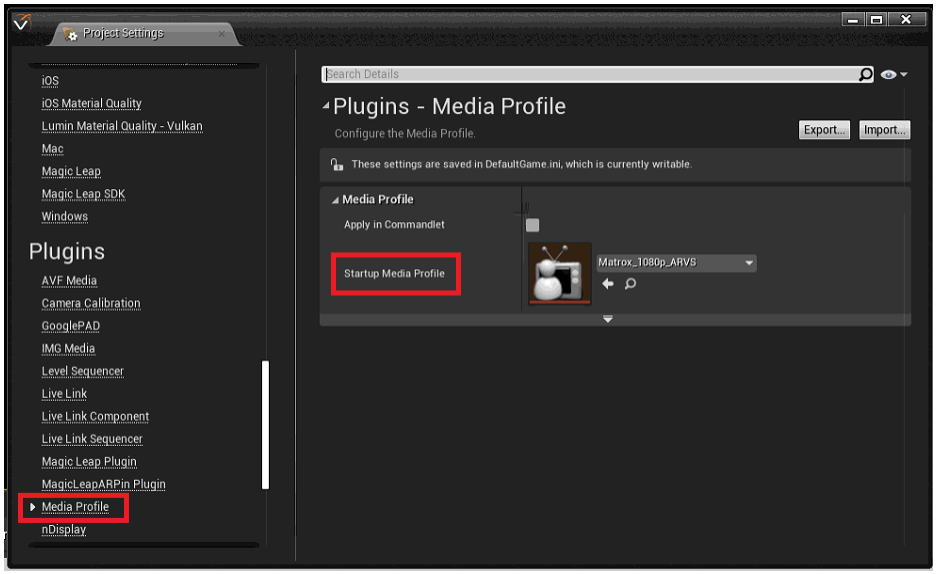
Continue with Selecting the Startup Media Profile.

Selecting the Startup Media Profile

If you want to run your project in **Game** mode (Voyager Node or nDisplay), you will need to select the **Startup Media Profile**.

To select the Startup Media Profile:

1. Select **Edit > Project Settings > Plugins > Media Profile**.
2. In the **Media Profile** plugin, from the **Startup Media Profile** drop-down, select the media profile you created for your project and then close the Project Settings editor.



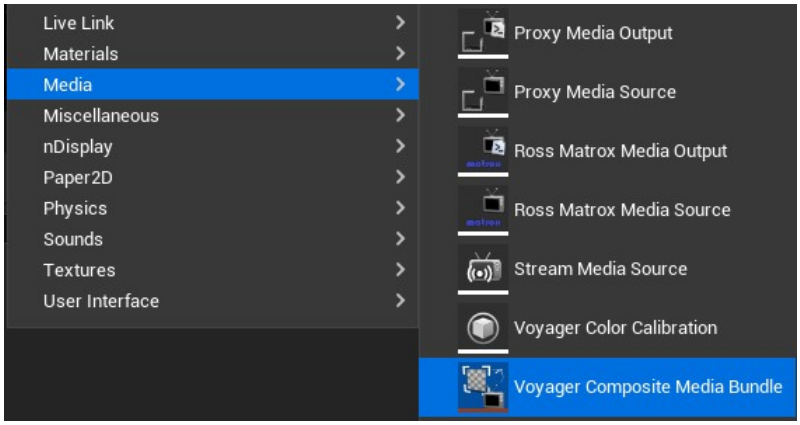
Select Startup Media Profile

Creating a Media Bundle

You need to create a media bundle asset for each input in your project. This will allow the asset to be controlled by other applications.

To create a media bundle:

1. Navigate to the **Voyager > LiveSources** folder.
2. Right-click in an empty section of the **Content** pane and select **Media > Voyager Composite Media Bundle** (for a composite input) or **Media > Media Bundle** (for a live source input).

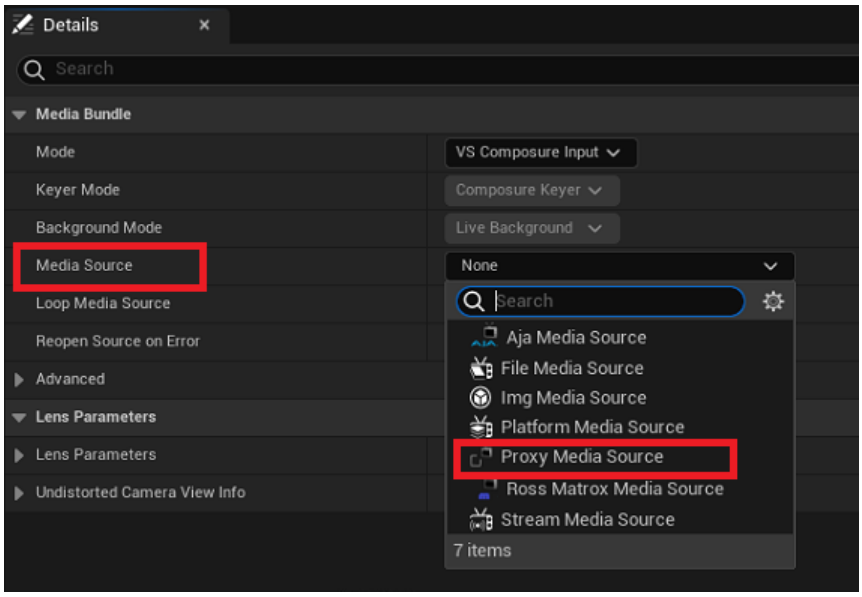


Create VoyagerComposite Media Bundle

3. In the **Content** pane, rename the **Media Bundle** asset to associate it with a composite input (e.g., **VoyagerComposite_1**) or a live source input (e.g., **LiveInput_1**).

   The **InnerAssets** folder is created automatically.

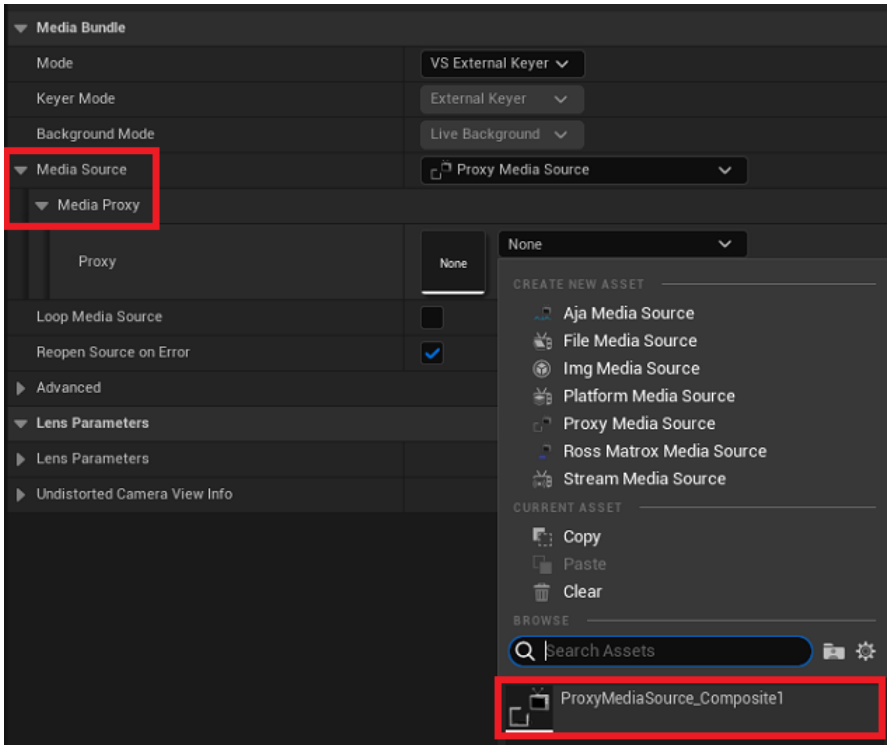
4. Double-click the media bundle you created to open the **Details** tab.
5. From the **Mode** drop-down, select the chroma keyer mode you need for your project.
6. From the **Media Source** drop-down, select **Proxy Media Source**.



Select Media Source

7. Expand **Media Source** and **Media Proxy** and from the drop-down, select the proxy media source to be

used with that media bundle.



Select Proxy Media Source

8. Repeat steps 2 to 7 for each input in your project.
9. Select **Save** and close the **Details** tab.

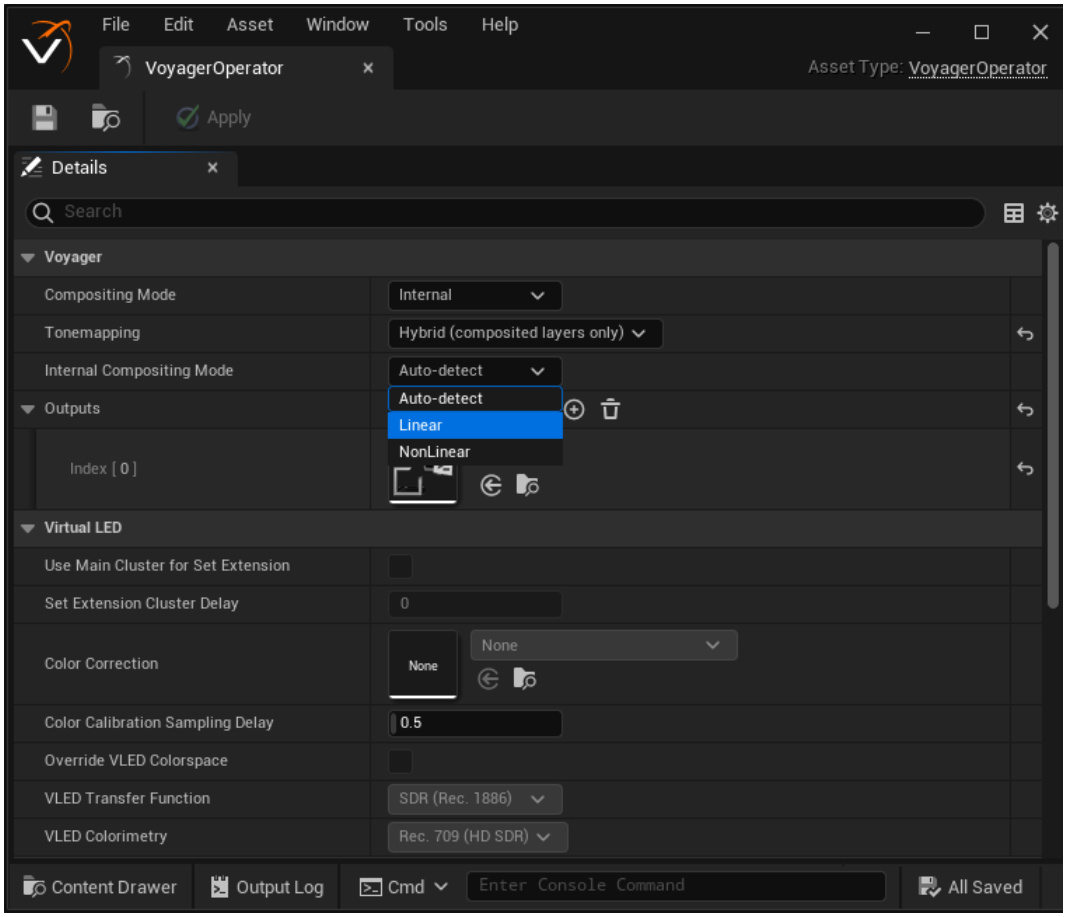
Creating and Configuring the Voyager Operator

The next step in the process is to create and configure the Voyager Operator.

To create and configure the Voyager Operator:

1. In the main toolbar, from the **Voyager** drop-down, select **New Voyager Operator**.
2. In the **Save Asset As** window, navigate to the **Voyager > Operator** folder and enter a name for the operator (e.g., **VoyagerOperator**) and select **Save**.

   The **VoyagerOperator** editor opens and the **VoyagerOperator_InnerAssets** folder is automatically created.



VoyagerOperator Editor

3. In the **VoyagerOperator** editor, in the **Voyager** section, from the **Compositing Mode** drop-down, select the appropriate option.

   The options are:

   - **Internal**
   - **External**
   - **Portal**
   - **nDisplay**

4. From the **Tonemapping** drop-down, select the **Tonemapper** you want to use.

   The options are:

   Native (broadcast tonemapper off) — The Unreal Engine native tonemapping and post-processing will be applied to everything in the level, including the incoming camera feeds. This setting is not recommended in internal compositing as the look of the camera feeds will be affected.

   Broadcast — A broadcast tonemapping (designed to preserve the look of the incoming camera feed) will be applied to everything in the level. Post-processing will affect everything in the level, including the incoming camera feeds

   Hybrid (composited layers only) — A broadcast tonemapping (designed to preserve the look of the incoming camera feed) will be applied only to the incoming camera feeds. Native tonemapping and post-processing will only affect the graphics, not the incoming camera feeds used for the composite layers. This is the recommended option.

   Hybrid (no color grading) — Similar to the **Hybrid (composited layers only)** option, but also disables the color grading post-processing options on the rest of the set, so that the broadcasting tonemapper is used everywhere. Unlike the **Broadcast** option, the other post-processing effects can still be applied to the rest of the set, while not affecting the composited layers.

5. From the **Internal Compositing Mode** drop-down, select from the following:

   Auto-detect — uses Linear compositing (engine default) unless the Voyager Media Bundle on the first compositing layer is set to **External Keying** and **Tonemapping** is set to **Hybrid**.

   Linear — recommended for multiple compositing layers, as it supports the most features and layering combinations.

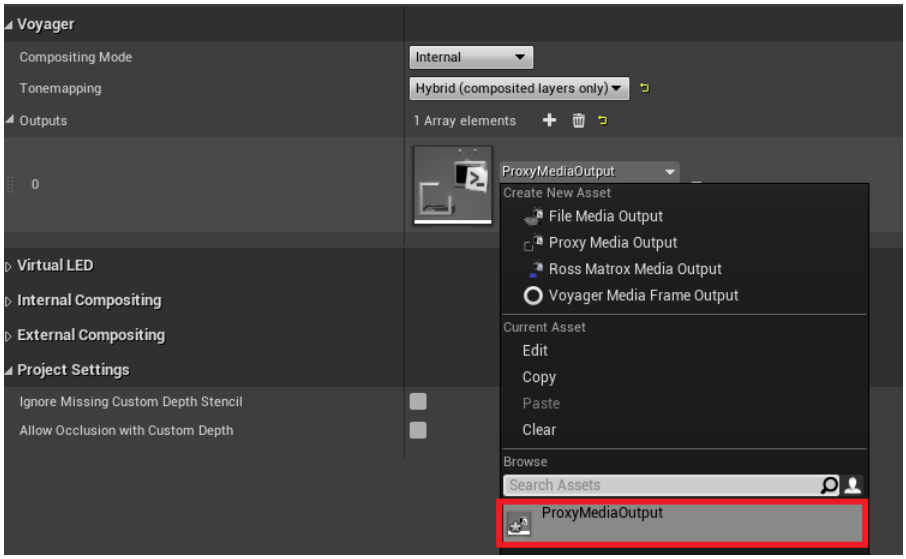
   Non-Linear — recommended for a single layer using external keying, as it supports fewer layering and post-processing combinations but replicates the external compositing typical of broadcast devices.

   ★In **Non-Linear** mode, pixels in the virtual graphics that are hidden by composite objects will still generate bloom, as if they were visible. This mode is intended for use with external keyers. It could have undesired effects on the edges of internal chroma keyers.

   ★Additional live sources that are fill-only without using chroma keying can be used in either mode, based on the nature of the main composite's keying method.

6. In the **Outputs** section, select the **+** icon to add one **Array element**.

7. Select the **Output** drop-down and scroll down to select the proxy output (**ProxyMediaOutput**) you created earlier.



Select Proxy Media Output

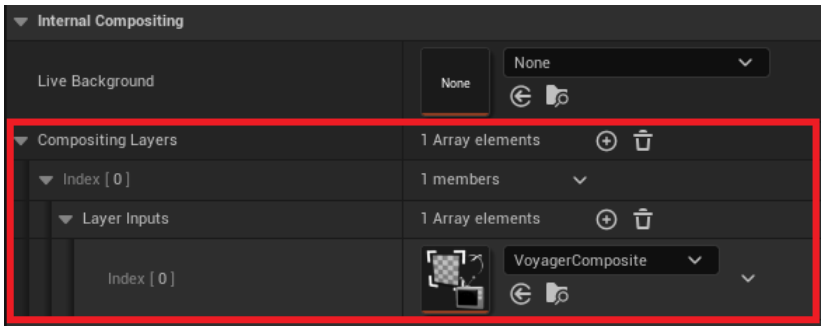
8. If you selected **External** in **Step 3**, in the **External Compositing** section, from the drop-down select whether you are using a **Virtual Set** or **Augmented Reality**. Then select **Save** and close the **VoyagerOperator** editor.

   OR

   If you selected **Internal** in **Step 3**, you'll need to add a **Voyager Composite Media Bundle**.

To add a Voyager Composite Media Bundle:

1. In the **Voyager Operator** editor, expand the **Internal Compositing** section.
2. Select the **+** icon in the section **Compositing Layers** to add an array element.
3. Expand **Index [0]** and add an array element to the **Layer Inputs** section.
4. From the **Layer Inputs** drop-down, select **Create New Asset > Voyager Composite Media Bundle**.



Add Voyager Composite Media Bundle

5. In the **Save Asset As** window, navigate to the **Voyager > Live Sources** folder and enter a name for the media bundle (e. g. **VoyagerComposite**) and select **Save**.

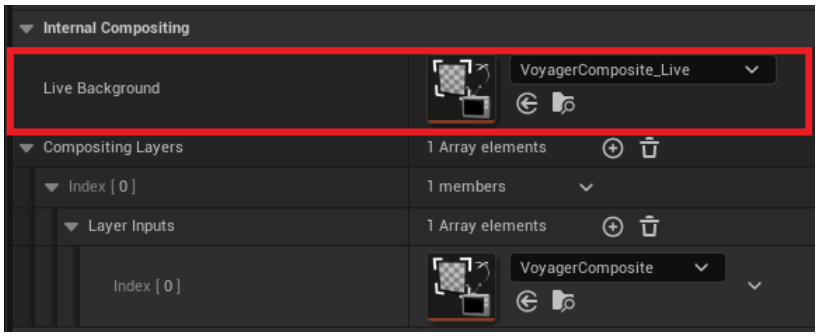
   The **VoyagerComposite_InnerAssets** folder is automatically created.

6. If you will not have a live background in your project, select **Save** and close the **VoyagerOperator** editor.

   If you do want a live background in your project, continue with To configure a live background.

To configure a live background:

1. Select the **Live Background** element drop-down and select **Create New Asset > Voyager Composite Media Bundle**.



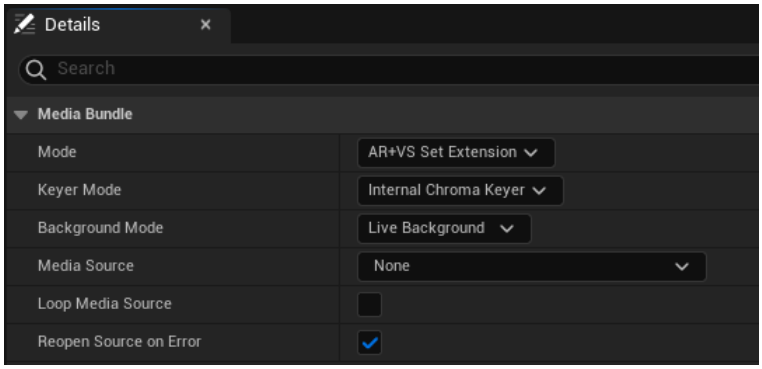
Voyager Composite Live Background

2. In the **Save Asset As** window, navigate to the **Voyager > Live Sources** folder and enter a name for the media bundle (e.g., **VoyagerComposite_Live**) and select **Save**.

   The **VoyagerComposite_Live_InnerAssets** folder is automatically created.

3. Then double-click on the **Live Background** icon and in the **Details** tab, from the **Mode** drop-down, select **AR+VS Set Extension**.

   This mode will work for either an augmented reality project or a virtual set project.



Voyager Composite - Live Source Details

4. From the **Keyer Mode** drop-down, select the chroma keyer mode you want to use, typically the **Internal Chroma Keyer**.
5. From the **Background Mode** drop-down, select **Live Background**.
6. Select **Save** and close the **VoyagerComposite_Live Details** tab.
7. In the **VoyagerOperator** editor **Details** tab, select **Save** and close the editor.

Continue with Creating and Configuring the Voyager Tracker.

Creating and Configuring the Voyager Tracker

The next step is to add a Voyager Tracker actor to the project and configure it as needed.

Service Location Protocol

Voyager supports the **Service Location Protocol** (SLP) which allows auto-discovery of the existence, location, and configuration of networked services. This feature is enabled by default, but if you want to use it, you need to ensure that **Port 427** is enabled.

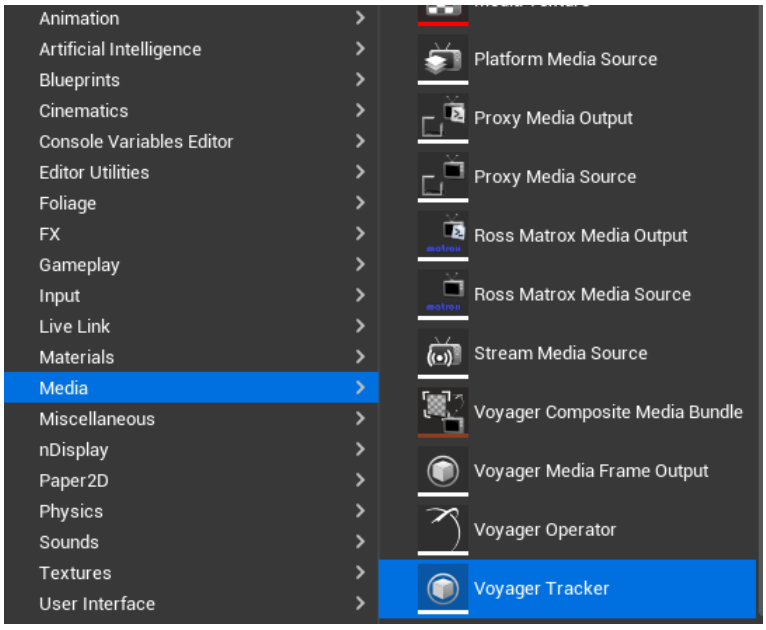
The following services can be found when SLP is enabled:

- Voyager (engine): service:voyager.rossvideo
- Voyager API: service:api.voyager.rossvideo
- Voyager Trackless: service:trackless.voyager.rossvideo
- Lucid: service:lucid.voyager.rossvideo
- Voyager Tracker: service:tracker.voyager.rossvideo

If you find that the networked services are not being discovered you can test the SLP using OpenSLP. See Appendix B: Testing SLP.

To create the Voyager Tracker:

1. In the **Content Drawer**, navigate to the **Voyager > Tracker** folder.
2. Right-click in the empty space and select **Media** and then **Voyager Tracker** at the very bottom of the menu and give it a name (e.g., VoyagerTracker).

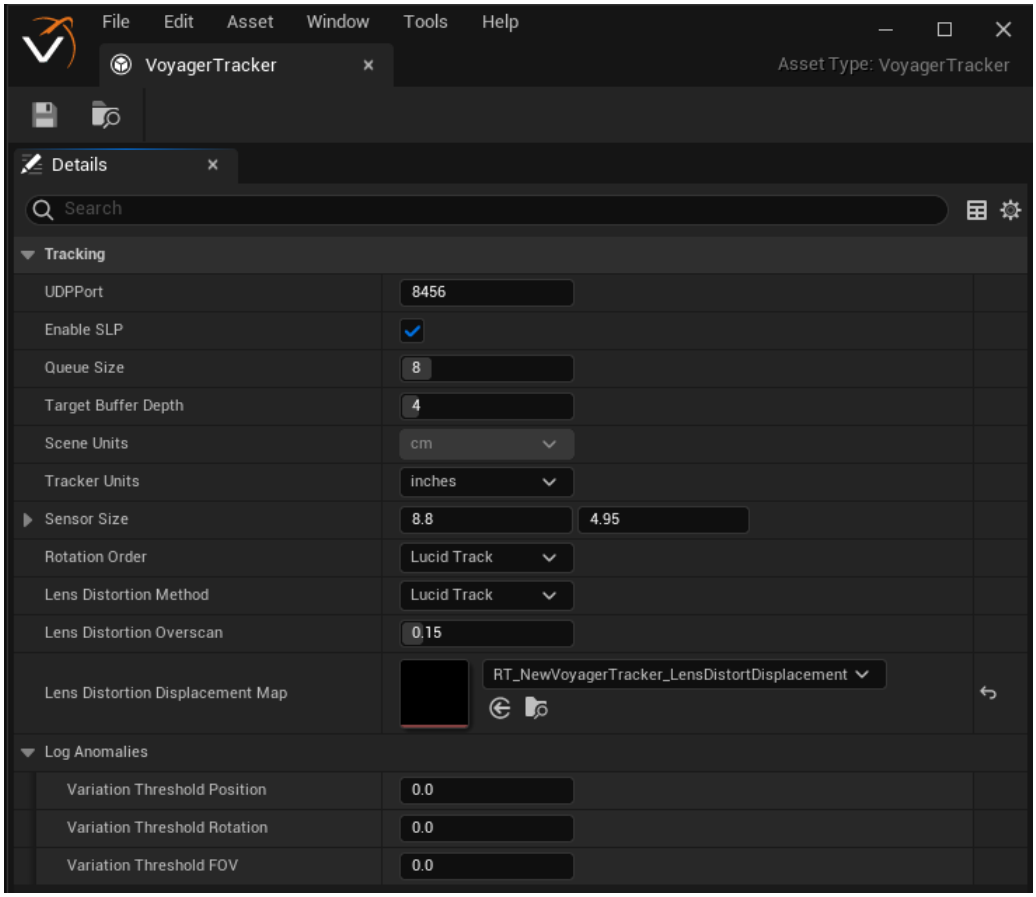


Add Voyager Tracker

The **VoyagerTracker_InnerAssets** folder is automatically created.

To configure the Voyager Tracker:

1. In the **Content** pane, double-click **VoyagerTracker** to open the **VoyagerTracker Details** tab.



VoyagerTracker - Details

In the **Details** tab, the **UDP Port** is by default, the same as the default port in Lucid Track.

If you need to change the port in either Voyager or Lucid Track, it needs to be changed in both.

2. The **Enable SLP** (Service Location Protocol) checkbox is selected by default. Clear the checkbox if you do not want to use it.
3. The **Queue Size** setting needs to be larger than the **Target Buffer Depth** to allow more tracking packets to be held in the queue and released when appropriate.

   The **Target Buffer Depth** setting is used to create a delay in the application of the tracking data, to align with the incoming video feed.

   Increasing the value increases the delay of the tracking data.
4. In the **Scene Units** field, leave the units at the default setting of **cm**.
5. In the **Tracker Units** field, ensure that the units selected match the units set in your renderer.

   The default **Sensor Size** values should work for most broadcast situations.

   The default **Rotation Order** values do not need to be changed.

6. From the **Lens Distortion Method** drop-down, select a lens distortion type as follows:
   - Select **Lucid Track** if you are using a lens distortion type from the list in Lucid Track.
   - Select **Spherical** if you are using a lens with spherical distortion curves.
   - Select **Overscan Only** if you are using the Portal effect.
   - Select **None** if you are not using lens distortion.
7. Leave the **Lens Distortion Overscan** setting at **0.15**.

   Increasing this value may cause performance issues.

   The **Lens Distortion Displacement Map** is automatically created when the Tracker is created and needs no change.
8. In the **Log Anomalies** section, enter or use the arrows to select a value for each of the following options. A value of "**0**" indicates that it is disabled.

   Variation Threshold Position — log variations in position between two tracking packets if beyond this threshold.

   Variation Threshold Rotation — log variations in rotation between two tracking packets if beyond this threshold

   Variation Threshold FOV — log variations in FOV between two tracking packets if beyond this threshold.
9. Select **Save** and close the **VoyagerTracker Details** window.

   Continue with Setting Up Compatibility With Voyager.

Setting Up Compatibility With Voyager

To make your project compatible with Voyager, you'll need to add a **Voyager Camera Actor** to the level. Then you will add the assets you created in the previous sections and configure them to connect with each other.

Which assets you add depends on whether you're doing internal compositing or external compositing. You will need to add a **Voyager Camera Actor** for either method.

- Internal Compositing: **VoyagerOperator**, **VoyagerTracker** and **VoyagerComposite**
- External Compositing: **VoyagerOperator** and **VoyagerTracker**

If you have more than one level in your project you need to add these assets to each level.

To add a Voyager Camera Actor to the scene:

1. If you haven't already opened the project level, open the project **Maps** folder and select on the level now to open it.
2. In the **Place Actors** tab, select the **Voyager** category and drag the **Voyager Camera Actor** into the level.

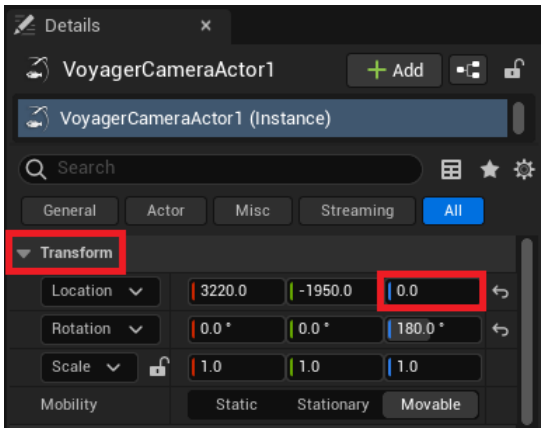
   By default the **Voyager Camera Actor** will be named **VoyagerCameraActor1**.

Place Voyager Camera Actor

3. For **External Compositing - AR** only:

   In the **Outliner**, delete any actors that aren't necessary (e.g., Atmospheric Fog, Floor, Player Start, Sky Sphere, SkyLight, SphereReflectionCapture).

4. In the **Outliner**, with **VoyagerCameraActor1** selected, in the **Details** tab, in the **Transform** section, make sure that the **Location Z** position value is set to **0.0** (or at the same level as the virtual floor if it is not at **0.0**).



Position the VoyagerCameraActor at Floor Level

5. Maneuver the **Voyager Camera Actor** in the scene using the **X** and **Y** arrows (one at a time) to the position needed for the view you want to show.

   The **Location** and **Rotation** values in the **Transform** section will adjust accordingly.

6. Press **Play** to verify the final location of the camera in the virtual world.

7. If you're not seeing what you expect to see, your camera actor may be turned the wrong way. Try changing the **Z Rotation** value (e.g., change it to **180.0**, if it is currently **0.0**).

To add assets:

- In the **Content Browser**:
  - Open the **Voyager > Operator** folder and then select and drag **VoyagerOperator** into the level.

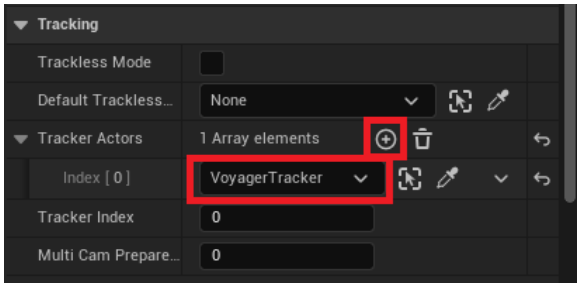
    The location of the **VoyagerOperator** actor in the scene is not important. It can be placed anywhere.
  - Open the **Tracker** folder and then select and drag **VoyagerTracker** into the level.

    The location of the **VoyagerTracker** actor in the scene is not important. It can be placed anywhere.
  - Open the **LiveSources** folder and then select and drag your **VoyagerComposite(s)** and live background (if using) into the level (for internal compositing only, these assets are not required for external compositing).

    If you are creating an AR or VS project with set extension, in the **Place Actors** list, select **Voyager** and then select either the **Voyager AR Composite (Background) Blueprint** or the **Voyager VS Tracked Composite (Keyer) Blueprint** and drag it into the scene. The selected actor will automatically be populated with the settings that were configured in the **VoyagerOperator** asset.
  - Scroll down to the **Voyager Content > Blueprints** folder and then select and drag the **FreeRoamingCamera** into the level, if you are using one.

To configure VoyagerOperator:

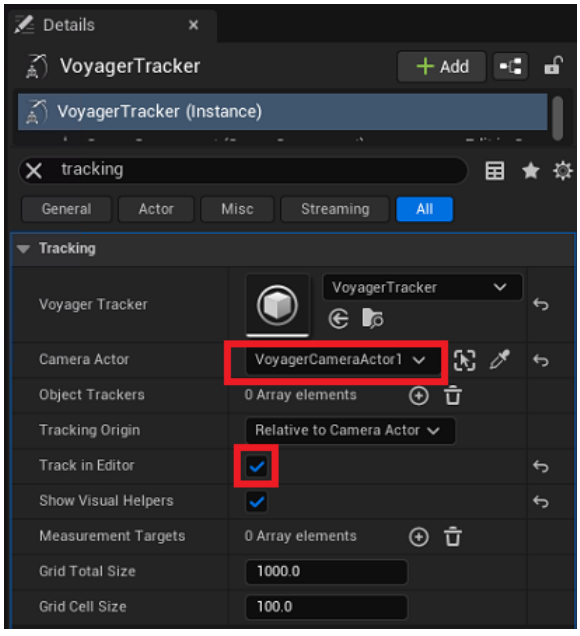
1. In the **Outliner** tab, select **VoyagerOperator**.
2. In the **Details** tab, in the **Tracking** tab, select the **+** icon to add one **Tracker Actor Array** element and from the drop-down for the new element, select **VoyagerTracker**.



Add Voyager Tracker Actor

To configure VoyagerTracker:

1. In the **Outliner** tab, select **VoyagerTracker**.
2. In the **Details** tab, in the **Tracking** tab, from the **Camera Actor** drop-down, select **VoyagerCameraActor1**.
3. From the **Tracking Origin** drop-down, select one of the following tracking origin positions:
   - Relative to Camera Actor
   - Relative to Tracker Actor
   - Absolute (Relative to World Origin)
4. Select the **Track In Editor** checkbox to enable camera tracking.

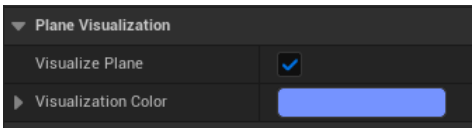


Configure VoyagerTracker

5. Select the **Show Visual Helpers** checkbox if you want to check that the units of distance between **VoyagerTracker** and the **VoyagerCameraActor** are correct.
   - If you are using external compositing, your project setup is now complete.
   - If you are using internal compositing, continue with the next section, **To configure the Voyager Composite actor**.

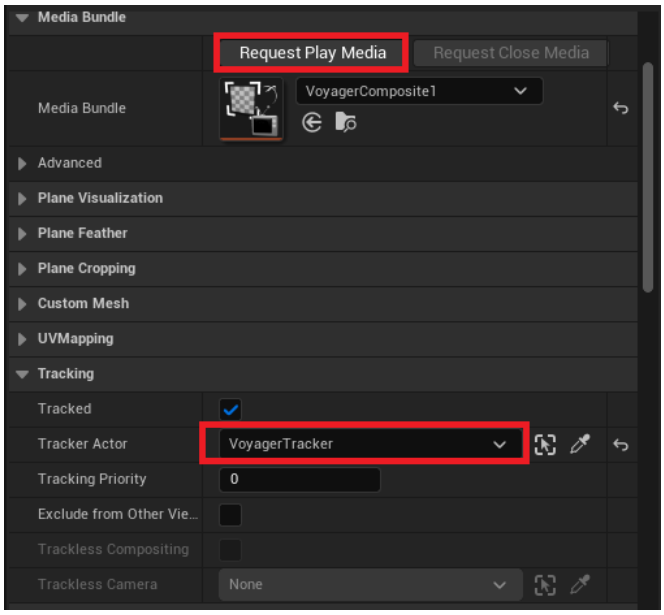
To configure the Voyager Composite actor:

1. In the **Outliner** tab, select **VoyagerComposite1**.
2. In the **Media Bundle** section, from the **Media Bundle** drop-down, select the **VoyagerComposite1** media bundle.
3. Make sure the front side of the **VoyagerComposite1** actor is facing you.
4. If you don't see a white face on the **VoyagerComposite1** actor:
   a. In the **Details** tab, in the **Plane Visualization** tab, select the **Visualize Plane** checkbox and select a color that will make it easy to see the plane in the viewport.



Plane Visualization

   b. In the **Transformation** section, use the **Y Rotation** value to rotate the **VoyagerComposite1** actor until you see the white face. For example, if the **Y Rotation** value is **0.0**, change it to **180.0**.
   c. Deselect the **Visualize Plane** checkbox.
5. In the **Media Bundle** section, select the **Request Play Media** button.



Configure VoyagerComposite1

6. In the **Tracking** section, from the **Tracker Actor** drop-down, select **VoyagerTracker**.

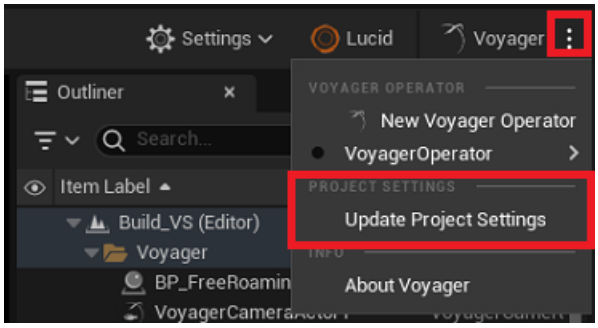
   Your project setup is now complete.

Updating Voyager Project Settings

Once you've set up your project, it's time to update the Voyager project settings.

To update the project settings:

1. Select the 3 vertical dots beside the Voyager icon and select **Select a Voyager Operator > VoyagerOperator**.
2. Then select **Update Project Settings**.



Updating Voyager Project Settings

3. In the **Voyager Project Defaults** dialog, select **Yes** to proceed.

Adding Voyager Color Calibration

If you are making an nDisplay with set extension project compatible with Voyager and want to ensure consistent colors across your displays, you can add and run the **Voyager Color Calibration** asset to your project. This is available in Voyager 5.1.1 and newer versions.

This process will sample multiple points in every dimension, comparing the colors in the AR part of the set with the colors in the video wall. A visual representation of the results is shown in the **Output** window.

Changes in lighting in the studio can affect the calibration, as can the LED calibration and camera settings, so if any of these things changes, the calibration would need to be redone. Once done, the calibration can be reused in other projects, as long as these factors remain the same.

The process takes about 15 minutes.

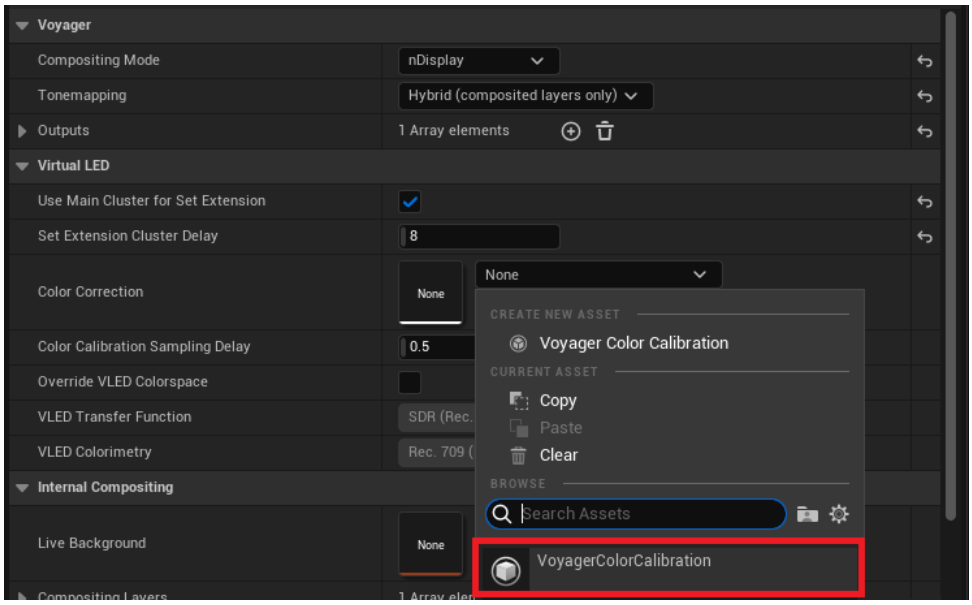
In a good calibration, most of the mesh will be blue.

- Blue mesh represents colors that were able to be reproduced.
- Green mesh represents colors that were mapped to the closest color possible.
- Orange mesh represents colors that are outside the sampling area.

If after calibration, the mesh is not mostly blue, you can make some adjustments and run the calibration again. See Adjusting the Voyager Color Calibration.

To add the Voyager Color Calibration asset:

1. In the **Outliner**, select the **VoyagerOperator** actor and in **Details > Voyager**, double-click the **VoyagerOperator** icon to open the editor.
2. In the **VoyagerOperator** editor, in the **Voyager** section, from the **Compositing Mode** drop-down, select **nDisplay**.
3. In the **Virtual LED** section, select the **Use Main Cluster for Set Extension** checkbox.
4. Also in the **Virtual LED** section, from the **Color Correction** drop-down, select **Voyager Color Calibration**.



Add Voyager Color Calibration

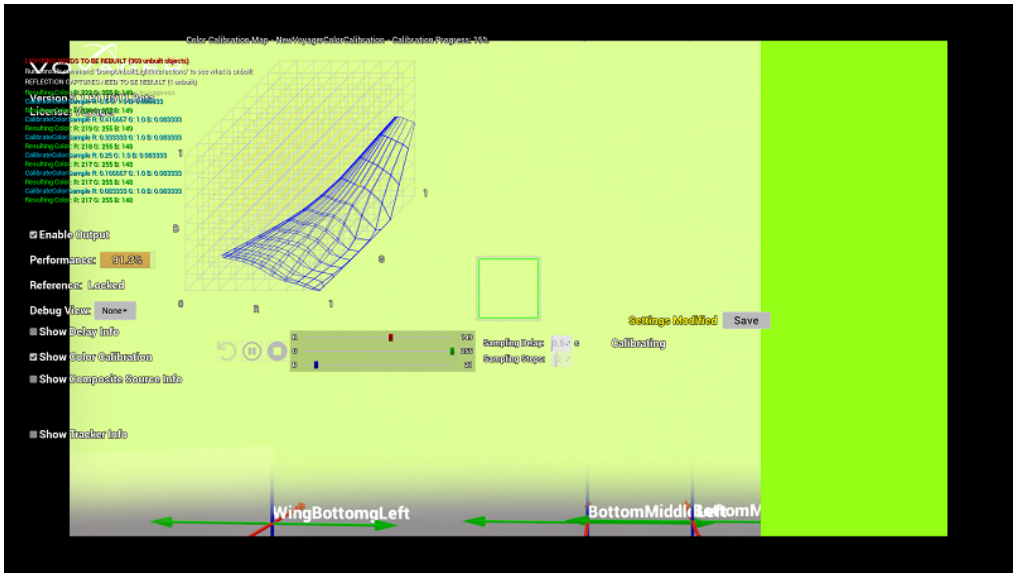
5. Select **Save** and in the **Apply Voyager Operator to Project** confirmation dialog, select **Yes**.

6. Close the **Voyager Operator** editor and in the main Voyager screen, select **Save** again and close the project.

To run the Voyager Color Calibration:

1. Launch your project using Voyager Switchboard Launcher.
2. In the **Preview** window, select the **Show Calibration** checkbox.

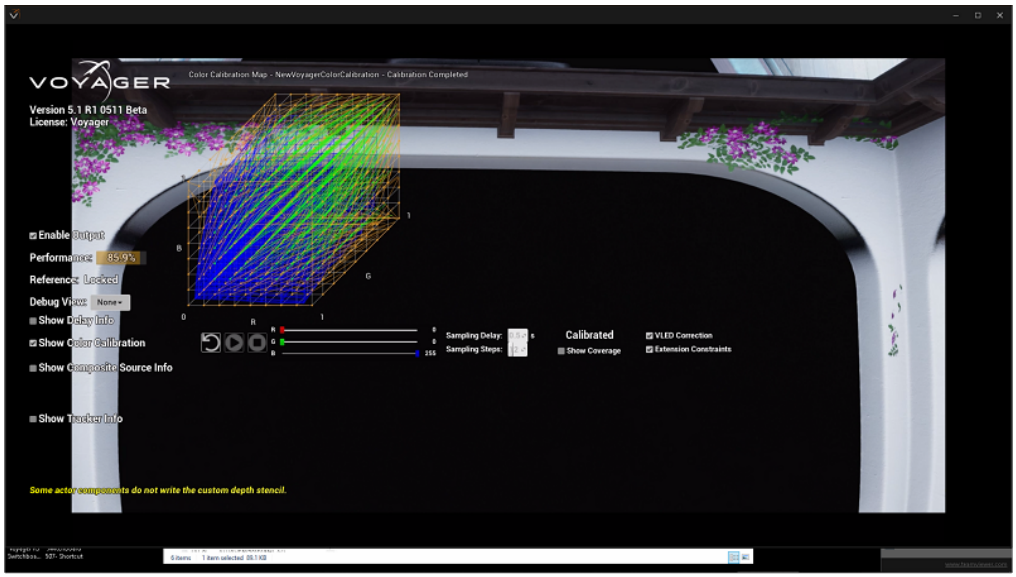
   The **Voyager Color Calibration** asset will begin sampling colors. During the calibration, the RGB sliders will show the color being calibrated, giving you an idea of the progress.



Color Sampling

3. Deselect the **Show Color Calibration** checkbox once calibration has started to save on engine performance while the system is calibrating.
4. When the calibration is finished, select the **Save** button.

   A good color calibration will show a mesh similar to the one in the image below:



Good Color Calibration

Testing the Calibration

When the calibration is completed and saved, you can use the RGB sliders to test the calibration.

To test the calibration:

1. Touch any of the **RGB** sliders to enter testing mode.
2. Move the slider to select a color to check the match between the AR extension and the LED wall.
3. Select the **Stop** button to exit testing mode.

Tips

If during the calibration process, a new color shows up on the AR side (the bar on the right) before the previous color has appeared in the sampling area (the box outline), increase the **Sampling Delay** value to give the calibration more time to sample each point and rerun the calibration.

Increasing the **Sampling Steps** value will increase the precision of the calibration but will also increase the time it takes to calibrate.

To run the calibration again:

1. Select the **Reset** button to reset and initialize the calibration and then select the **Start** button to start a new calibration.

Calibration Reset and Start

2. When the calibration is finished, select the **Save** button.

Organizing Voyager Assets

It's helpful from an organizational standpoint to create a **Voyager** folder in the **Outliner** to contain the **Voyager** assets.

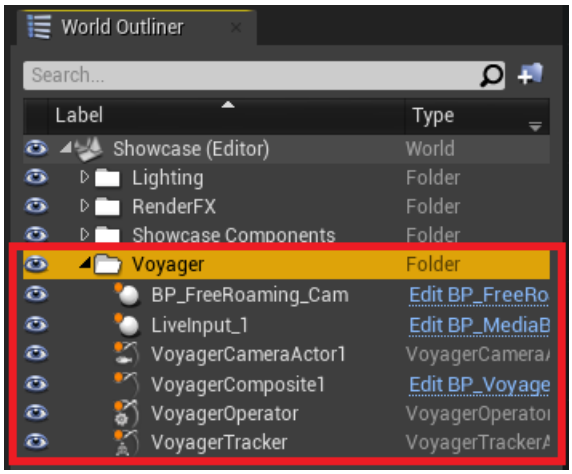
To create a Voyager folder in the Outliner:

1. In the **Outliner**, select one of the Voyager assets you created.
2. Select the **+** symbol to the right of the **Search** field.

   A new folder is created and the selected asset is placed inside the folder.
3. Name the new folder **Voyager**.
4. Right-click on each Voyager asset and at the bottom of the context menu, select **Move To**.

   Alternatively, you can press **Shift** and select to select multiple assets and move them all at once.
5. From the folder list, select the **Voyager** folder you just created.

   If you have a long list of folders, start typing "`Voyager`" in the **Search** field to find the **Voyager** folder quickly.



Voyager Folder in Outliner

Using Multiple Composite Inputs in Your Project

You can have multiple media sources playing in your virtual set at the same time.The composite planes on which the media sources will be displayed can be on a single layer or on up to 3 separate layers. You can display the same media source on multiple composite planes or have a different media source on each plane. The total number of composite planes you can have will depend on the size and complexity of your project.

There are some considerations to keep in mind when using multiple composites and multiple layers:

- Having only one composite on a layer is less costly in terms of performance, than having several composites on a layer; so if you have more than one composite, it's best to put them on separate layers.
- Each additional layer adds to the performance cost.
- Using a custom mesh for a composite is more costly than using the default mesh; so if you do want to use a custom mesh, it's best to put it on the same layer as another composite.
- Although in most situations, the compositing layers can be in any order, it is recommended that you place the compositing layers in the same order as they appear in the scene (starting from the back), whenever possible, particularly if there are some objects with translucent materials in between the compositing layers.
- If more than 2 compositing layers are overlapping in regions where they are all semi-translucent, the results may not be accurate.
- Voyager supports having a clean live source composite with up to two keyed talent trackless composite planes side by side on top of it. Performance may vary, depending on the configuration.

Adding Composite Inputs

If you are using one of the Voyager project templates, there will be one composite array element with one layer and one input already set up.

You will need to add a **VoyagerCompositeMediaProxy** for each composite plane you intend to have in your project. See Creating a Media Source Proxy for instructions.

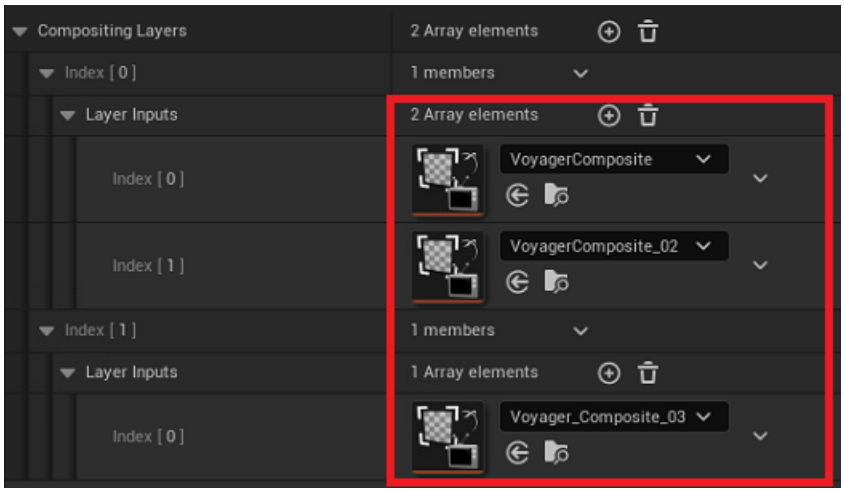
If you don't already have a media profile set up, you will need to create one. See Creating a Media Profile for instructions.

Prior to adding composite inputs, you will need to add a **VoyagerCompositeMediaBundle** for each composite plane you intend to have in your project. See Creating a Media Bundle for instructions.

To add composite inputs:

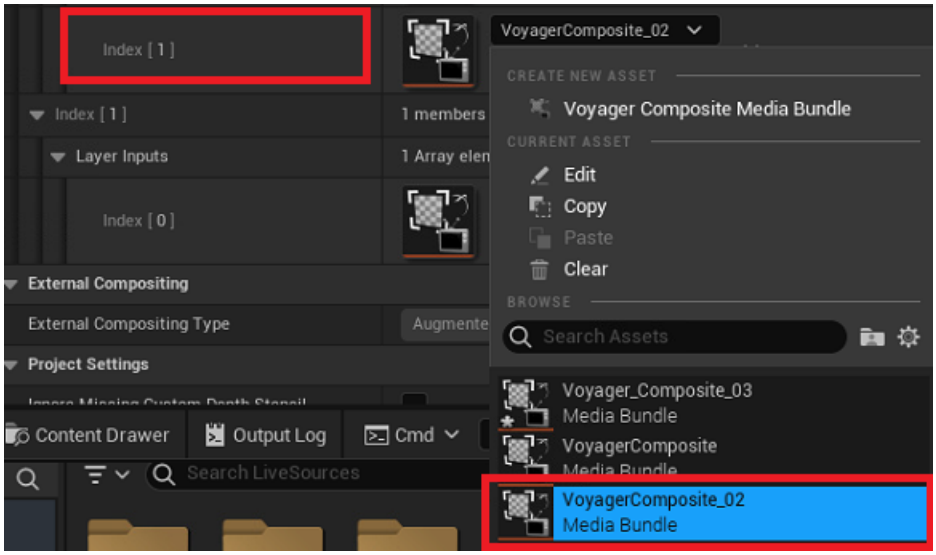
1. In the **Outliner**, select the **VoyagerOperator** actor.
2. In the **VoyagerOperator Details** tab, scroll down to the **Voyager** section and double-click the **VoyagerOperator** icon to open the **Details** editor.
3. In the **VoyagerOperator Details** editor, in the **Internal Compositing** section, expand the **Compositing Layers** section.

   If you are adding composite inputs to a Voyager template, **Layer 1** (**Index 0)** will already have 1 composite configured.
4. If you want to have composites on more than one layer, select the **+** icon beside the **Compositing Layers Array** elements to add another layer (called **Index [1]**).
5. In any layer, in the **Layer Inputs** line, select the **+** icon to add a composite input.



VoyagerOperator - Add Composite Inputs

6. Then from the **Layer Input** drop-down, select the **Media Bundle** for the input source you want to use.



Select Media Bundle

7. Repeat steps 5 and 6 to add as many composite inputs on each layer as you need.
8. Now you can add a **Composite Actor** for each composite you added in the above steps.

   See Adding the Composite Actor(s) to the Set.

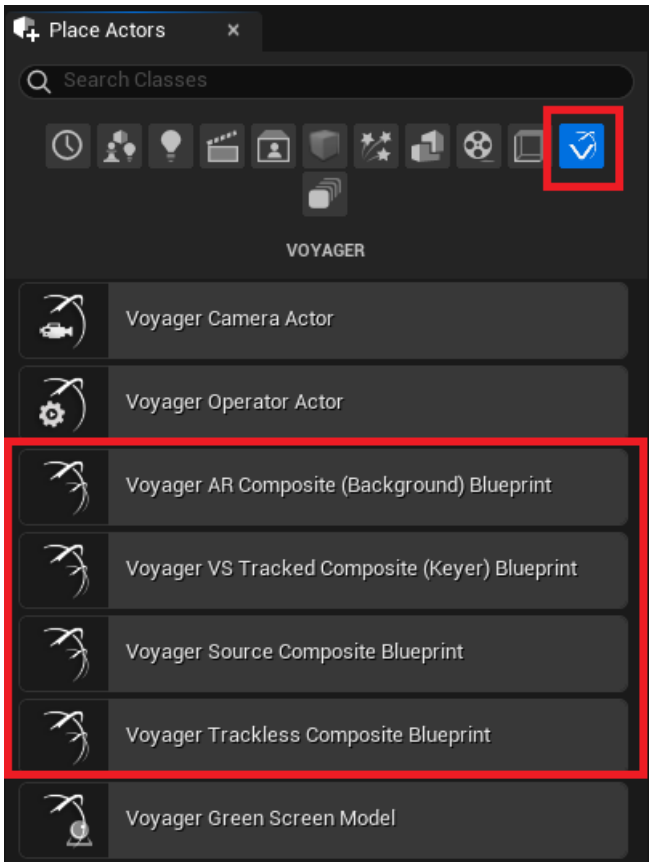
Adding a Composite Actor to the Set

You need to add a composite actor to your set for any additional composite inputs you've configured and assign a media bundle to each composite.

You may want to exclude one or more composite planes from the view of other cameras, perhaps when showing only a portion of the camera feed on one plane. In this case, you'll need to add a camera actor for the composite plane you want to exclude and assign the camera to that composite. See To exclude a composite plane from the view of other cameras.

To add composite inputs to the set:

1. In the **Place Actors** tab, from the **Voyager** category, select one of the following actors for each composite in your project:
   - Voyager AR Composite (Background) Blueprint
   - Voyager VS Tracked Composite (Keyer) Blueprint
   - Voyager Source Composite Blueprint (for use on a video wall)
   - Voyager Trackless Composite Blueprint (for use on a composite plane; available only in the Trackless template)



Select Composite Actor

2. Drag the blueprint into the level and position it where you want it.
3. In the **Outliner**, right-click the first **BP_MediaBundleActor** you brought in and rename it to something meaningful.

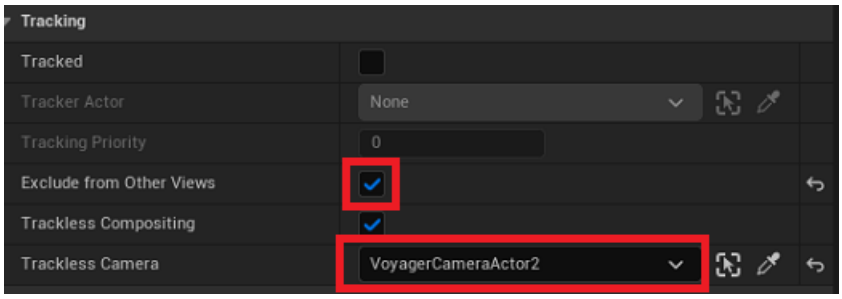
4. In the **Details** tab, select the **All** button and in the **Media Bundle** section, from the **Media Bundle** drop-down, select the media bundle you want to assign to the composite.

   The composite actor you drag into the level will automatically pick up the first media source that has been set up in the VoyagerOperator, but you may want to change this.

5. Repeat steps 3 and 4 for each composite you brought into the level.
6. Select **Save Current**.

To exclude a composite plane from the view of other cameras:

1. In the **Place Actors** tab, from the **Voyager** category, select and drag a **Voyager Camera Actor** into the level.
2. Position the **Voyager Camera Actor** to point towards the composite you want to exclude from other views.
3. In the **Outliner**, select the composite you want to exclude.
4. In the **Details** tab, in the **Tracking** section, select the **Exclude from Other Views** checkbox.



Voyager Composite - Exclude from Other Views

5. Then, from the **Trackless Camera** drop-down, select the camera actor you added to assign it to this composite.

Creating Live Sources

This section describes the procedure for creating a live video source that can be displayed on a surface in the scene. The procedure is the same whether you're using **Internal** or **External** compositing.

If you want to create multiple live sources that are "clean" (they have no TAA, post-processing, broadcast tonemapper, etc.), refer to Using Multiple Composite Inputs in Your Project.

The steps for this process are:

Configuring a Proxy Media Source for a Live Input

Configuring a Live Input

Creating a Media Bundle for a Live Source

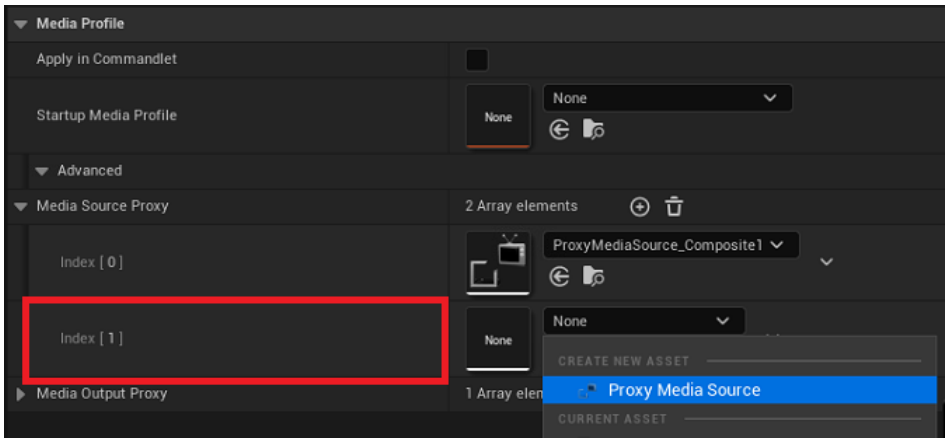
Creating a Live Source Material

Configuring a Proxy Media Source for a Live Input

To be able to add one or more live inputs to your project, you'll first need to add an equal number of media source proxies.

To add a proxy media source for a live input:

1. In the main menu, select **Edit > Project Settings** and scroll down to the **Plugins** section.
2. Select **Media Profile**.
3. Expand **Advanced** and then expand the **Media Source Proxy** section.
4. Select the **+** icon to add an **Array** element (**Index [#]**).



Proxy Media Source for a Live input

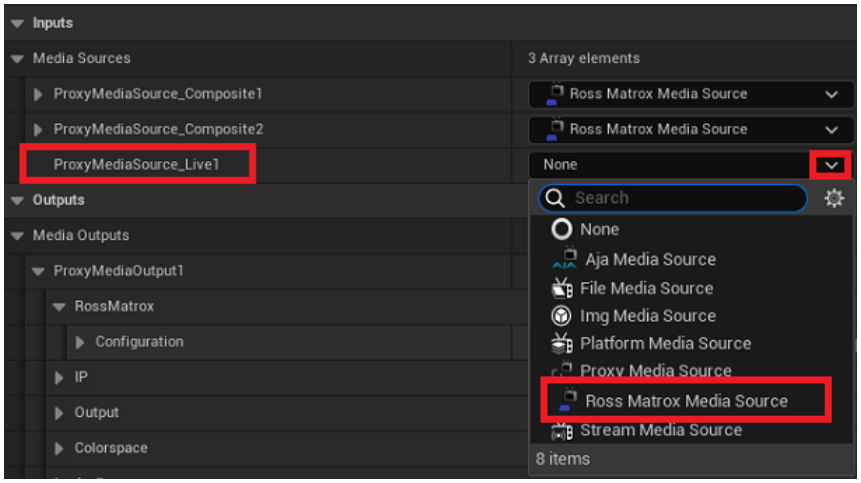
5. From the new element drop-down, select **Create New Asset > Proxy Media Source**.
6. In the **Save Asset As** window, navigate to the **Voyager > Proxies** folder and in the **Name** field, enter a name for the proxy (e.g., **ProxyMediaSource_Live1**).
7. Select **Save** and close the **Project Settings** window.

Configuring a Live Input

You can have several live inputs displayed on various surfaces in the project scene.

To configure a live input:

1. In the main toolbar, double-click the **Media Profile** icon to open the media profile you created in an earlier step.
2. In the **Details** editor that opens, expand the **Inputs > Media Sources** section, then select the **ProxyMediaSource_Live1** drop-down and select **Ross Matrox Media Source**.



Live Media Source Input

3. Expand the **ProxyMediaSource_Live1** input and then expand **RossMatrox** to access **Configuration**.
4. Select the **Configuration** drop-down and in the configuration panel that opens, select the options described below and select **Apply**.

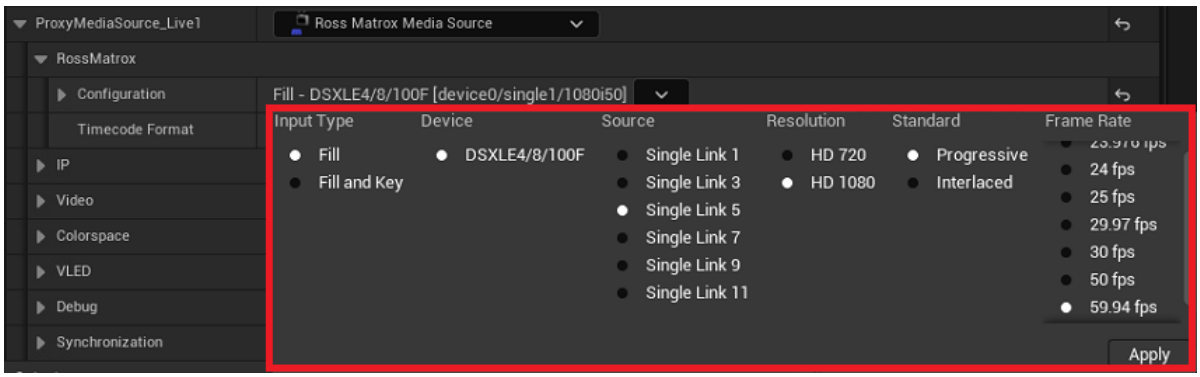
   The **Resolution** and **Frame Rate** options will differ depending on your hardware configuration.

   Input Type: Fill

   Device: DSXLE4/8/100F

   Source: Single Link 3 (or any pin not being used by another media source)

   Resolution/Standard/Frame Rate: The video formats that correspond to your workflow.



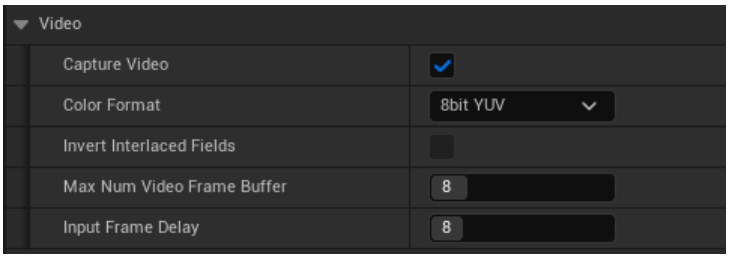
Input Configuration - Live Input

5. Expand **Video** and configure the settings as follows:
   - Select the **Capture Video** checkbox.
   - From the **Color Format** drop-down, select the color format that works best for your project.

     10bit YUV 422 is recommended when using **HDR** and **Wide Color Gamut**.

     With multiple inputs, there may be a performance cost when using **10bit YUV 422**. In this case, try using **8bit YUV 422**.

     10bit YUV 422 is not supported in a **Fill and Key** configuration with an AJA card. In this case you could use **RGBA**.
   - From the **Max Num Video Frame Buffer** drop-down, select **8** if you are using a **Progressive** format or if you are using an Interlaced format, select **1** or **16**.
   - From the **Input Frame Delay** drop-down, select the frame delay that works best for your project.

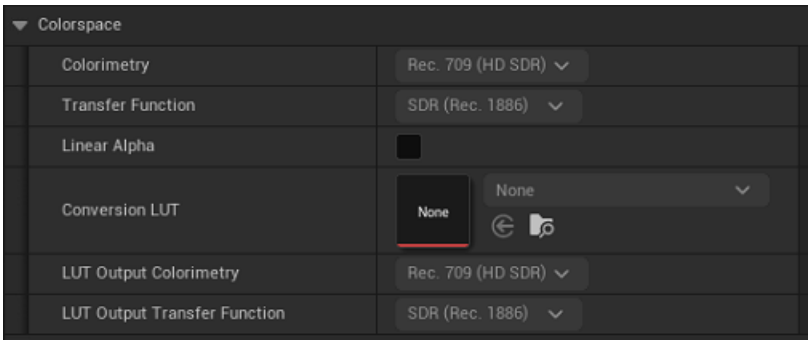


Video Configuration - Composite Input

6. Expand **Colorspace** and configure the settings as follows:
   - From the **Colorimetry** drop-down, select one of the following options:
     - **Rec. 709 (HD SDR)** for High Dynamic Range (increased levels in the range between bright and dark) and Standard Dynamic Range

       OR

     - Rec. 2020 (WCG) for Wide Color Gamut (increased selection of color values)
   - From the **Transfer Function** drop-down, select one of the following options:
     - **SDR (Rec. 1886)** — Standard Dynamic Range
     - **HLG (Rec. 2100)** — increases the dynamic range of the video and is compatible with both SDR and HDR displays
     - **HDR10 (PQ 1000 nits)** — supports a significantly larger range of brightness as SDR, with a corresponding increase in contrast and a color palette of one billion shades.



Colorspace Settings

- Select the **Linear Alpha** checkbox if the incoming alpha is already linear; the **Transfer Function** will not be applied.

  Refer to the documentation for your chroma keyer or key source to determine whether or not the alpha is linear.

- If you selected an 8bit color format in the **Video** settings, these are the only **Colorspace** settings available. Select **Save** and continue with the output configuration.
- If you selected a 10bit color format in the **Video** settings, the **HDR** settings will also be available. Continue with the next steps to edit **HDR** settings.
  - From the **Conversion LUT** drop-down, browse to and select the **Look Up Table** you want to apply to your color grading.
  - From the **LUT Output Colorimetry** drop-down, select either **Rec. 709 (HD SDR)** or **Rec. 2020 (WCG)**.
  - From the **LUT Output Transfer Function** drop-down, select either **SDR (Rec. 1886)**, **HLG (Rec. 2100)** or **HDR10 (PQ 1000 nits)**.

7. If you are configuring a live input for a **Virtual LED + Set Extension** project, expand **VLED** and select the **Activate on Set Extension Node Only** checkbox.

   For all other project types, leave this checkbox clear.

8. Repeat steps 1 to 7 for each **Live Input** in your project.
9. Select **Save** and close the **Media Profile** editor.

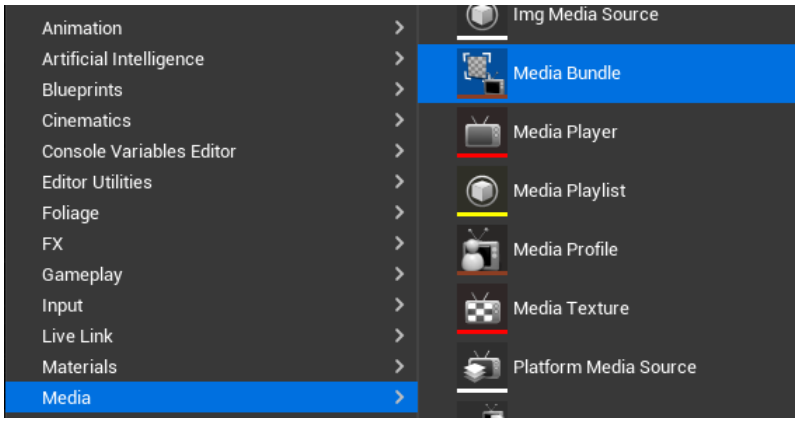
Creating a Media Bundle for a Live Source

You need to create a media bundle asset to play the video that you want to see on a surface in your scene. This will allow the asset to be controlled by other applications.

To create a media bundle:

1. Navigate to the **Voyager > LiveSources** folder.
2. Right-click in an empty section of the **Content** pane and select **Media > Media Bundle**.

   ★Do not select the **VoyagerCompositeMediaBundle** in this step.

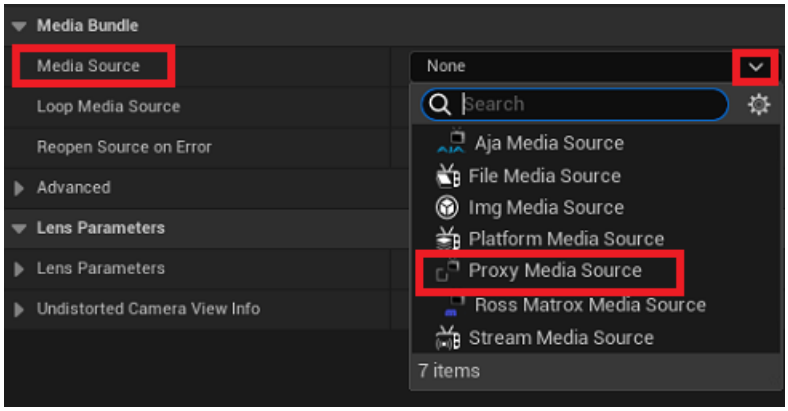


Create Media Bundle

3. In the **Content** browser, rename the **Media Bundle** asset to **LiveInput_1**.

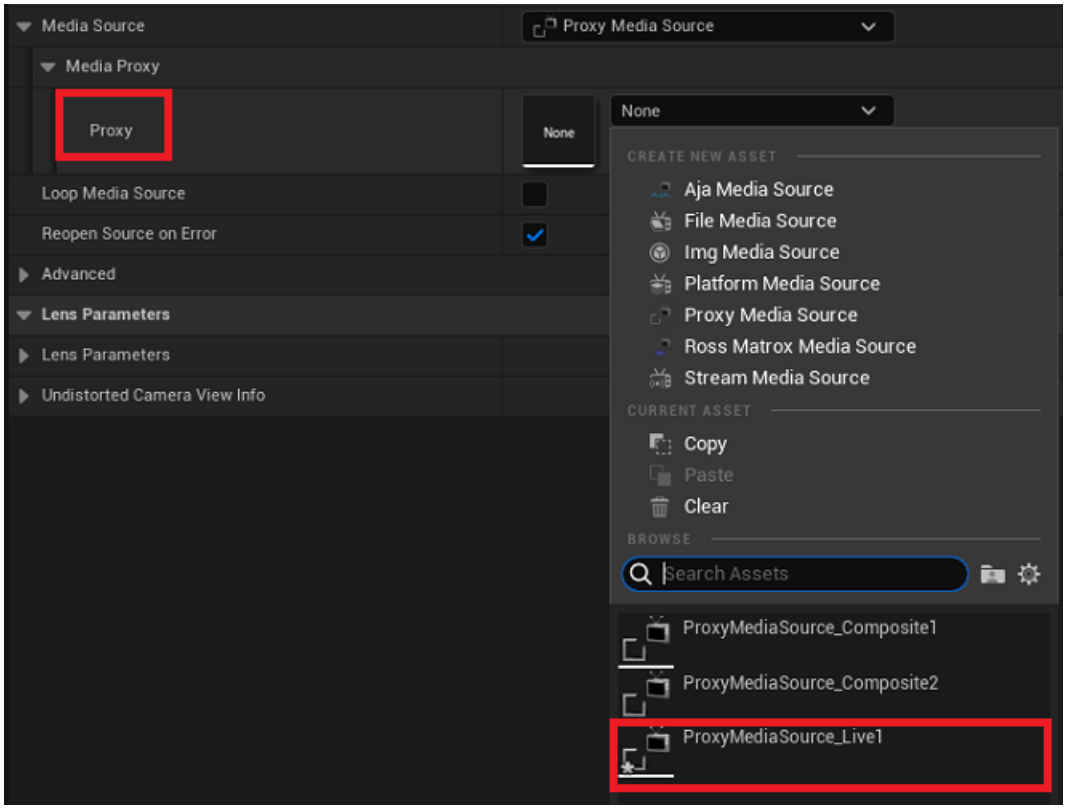
   The **LiveInput_1_InnerAssets** folder is created automatically.

4. Double-click the **LiveInput_1** media bundle to open the **Details** editor.
5. In the **Details** editor, from the **Media Source** drop-down, select **Proxy Media Source**.



Select Proxy Media Source

6. Expand **Media Source** and **Media Proxy** and from the **Proxy** drop-down, select **ProxyMediaSource_Live1**.



Select ProxyMediaSource_Live1

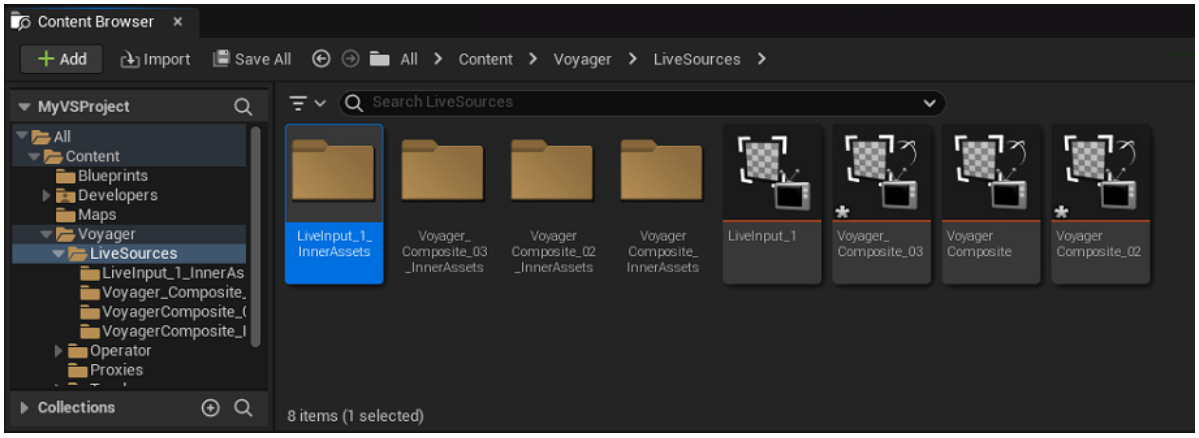
7. Select **Save** and close the **Details** editor.

Creating a Live Source Material

This procedure creates a material for the media bundle you created previously, using a texture present in the **Inner Assets** folder. It then hides the media bundle from the camera view.

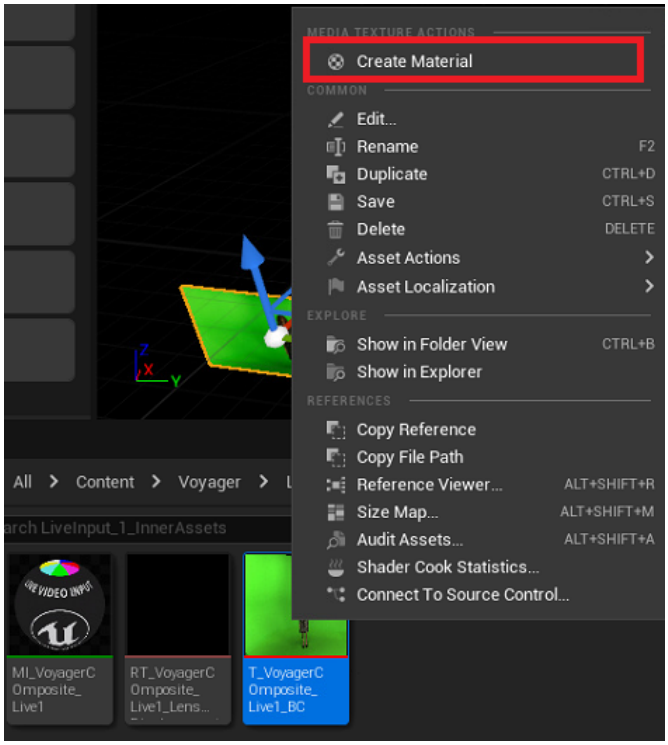
To create a live source material:

1. Drag the **LiveInput_1** media bundle into the level and position this actor so that it is out of sight of the camera (beneath the floor or elsewhere in the scene).
2. Double-click the **LiveInput_1_Inner Assets** folder to open it.



Open LiveInput_1_InnerAssets Folder

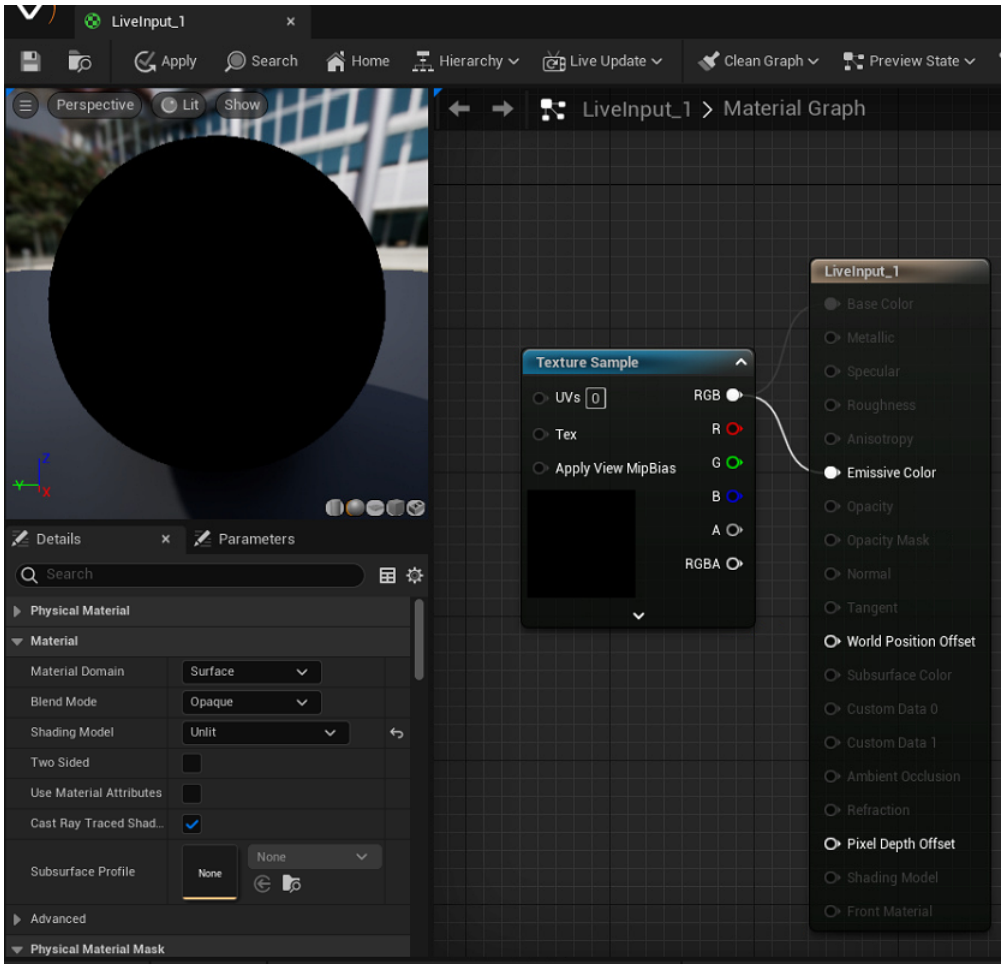
3. Then right-click on the **T_LiveInput_1_BC** media texture and from the context menu, select **Create Material**.



Select Create Material

4. In the content browser, name the material **LiveInput_1**.

5. Double-click the **LiveInput_1** material and in the **Details** tab, in the **Material** section, set the **Shading Model** to **Unlit**.



LiveInput_1 Material Details

6. In the blueprint, drag the **RGB** pin of the **Texture Sample** node to the **Emissive Color** pin of the **LiveInput_1** node.
7. Select **Save** and close the editor

Converting a Tracked Project to a Trackless Project

You can convert an existing Voyager tracked-camera project to a **Voyager Trackless Studio** project. This requires a few changes to the VoyagerOperator actor (if there is one in your project) and the composite(s), as described in this section. To keep your project clean, you will also be able to remove a few actors that will no longer be needed.

★This option is available starting in Voyager version 4.27 R3.

★Before starting this process, ensure the Voyager Trackless Plugin is installed and enabled in Voyager. See Enabling the Voyager Plugins.

Perform the following steps to convert your tracked project to a trackless project:

Modifying the VoyagerOperator

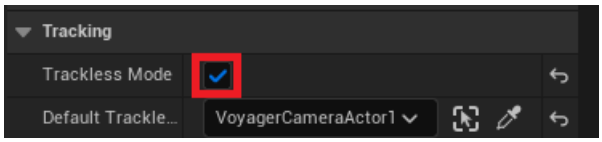
Configuring the VoyagerComposite(s)

Modifying the VoyagerOperator

In a tracked Voyager project, the VoyagerOperator is associated with the VoyagerTracker. In a trackless project, there is no VoyagerTracker. Instead, the default trackless camera is associated with the **VoyagerCameraActor**, as described in the following procedure.

To modify the VoyagerOperator:

1. Launch your tracked Voyager project.
2. In the **Outliner**, select the **VoyagerOperator**.
3. In **VoyagerOperator > Details > Tracking**, do the following:
   - Select the **Trackless Mode** checkbox.
   - Set the **Default Trackless Camera** to the corresponding **VoyagerCameraActor**.



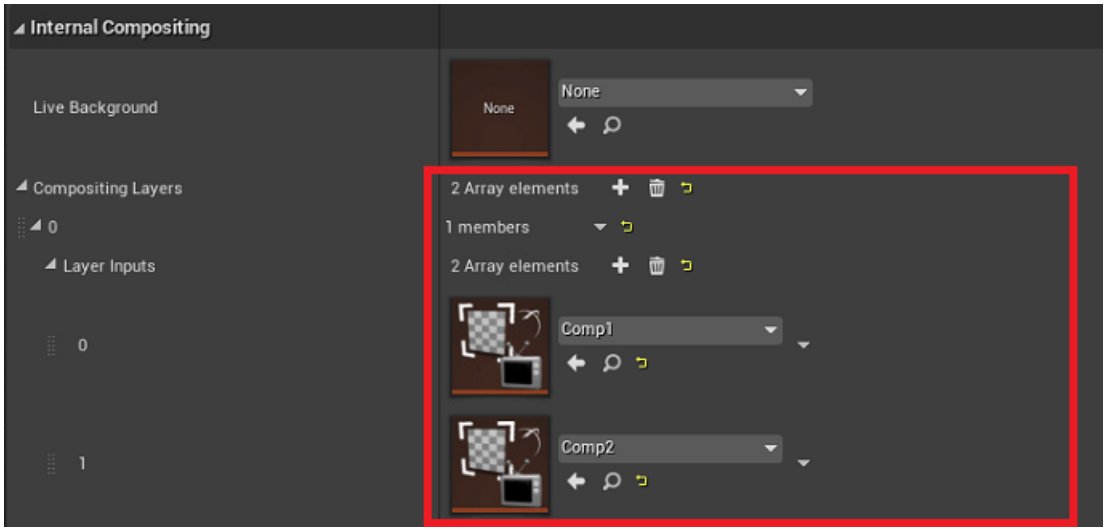
Select Trackless Mode

4. In the **Voyager** section, double-click the **VoyagerOperator** icon to open the **Details** editor.
5. In the **Details** editor, in the **Internal Compositing** section, check that your **Composite Media Bundle(s)** are listed in the **Composite Layers Array**.

   If there are no media bundles present, you will need to add one for each composite plane in your project. See Creating a Media Bundle for instructions.

   You will also need to add a **Media Source Proxy** for each composite. See Creating a Media Source Proxy.

   For information on using multiple composites on one or more layers, see Configuring Multiple Composite Inputs.



Add Composite Media Bundles

6. Select **Save** and close the editor.

Configuring the VoyagerComposite(s)

When converting a tracked project to a trackless project, you'll need to modify some settings in the **VoyagerCompositeActor(s)** and add a new component called **Always Face Camera**. There are 2 methods for doing this:

Method 1 — Make changes in the existing composite actor(s).

Method 2 — Replace each existing composite actor with a Voyager Trackless Composite.

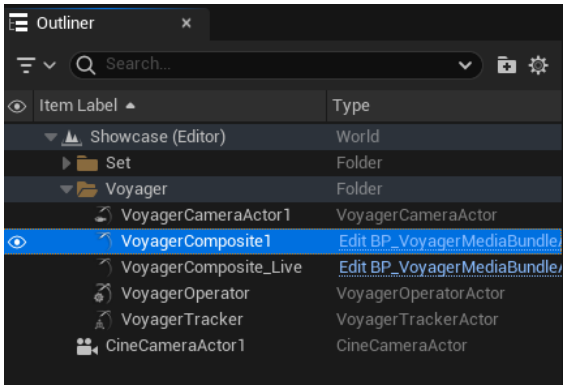
Both methods are described in this section.

Method 1

In the first method, you will modify existing composite actors.

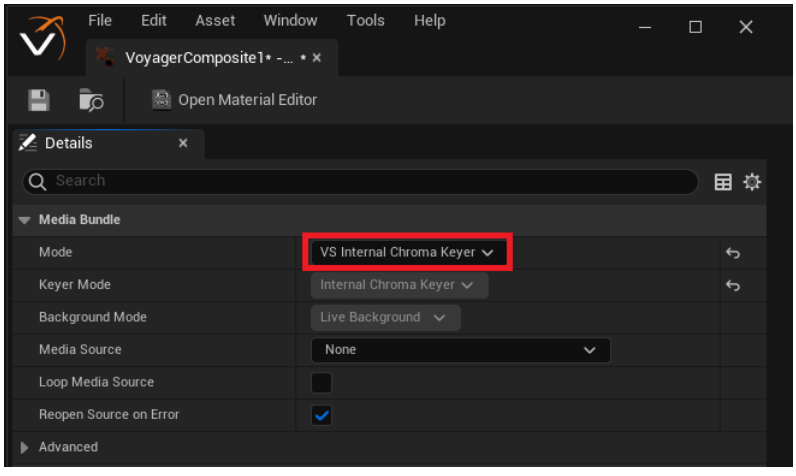
To modify a composite actor:

1. In the **Outliner**, select a composite actor.



Select a Voyager Composite Actor

2. In **Details > Media Bundle**, double-click the **Media Bundle** icon to open the composite editor.
3. In the composite editor, set the **Media Bundle Mode** to **VS Internal Chroma Keyer**, **VS External Keyer** or **VS Composure Input**, as appropriate for your chroma keyer setup.



Media Bundle Details - Select Mode

4. Select **Save** and close the editor.

5. In **VoyagerComposite > Details > Tracking**, make the following changes:

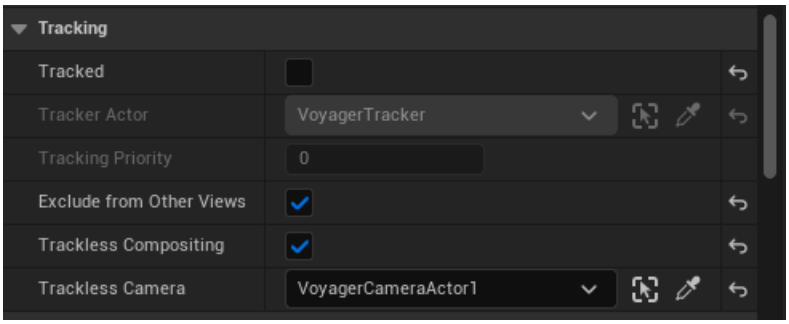
   a) Clear the **Tracked** checkbox.

   b) Select the **Trackless Compositing** checkbox.

6. If you have more than one composite and want to make sure that the camera for this composite doesn't see the other composite(s), do the following:

   a) Select the **Exclude from Other Views** checkbox.

   b) From the **Trackless Camera** drop-down, select the camera actor associated with the composite.



Trackless Mode - Exclude from Other Views

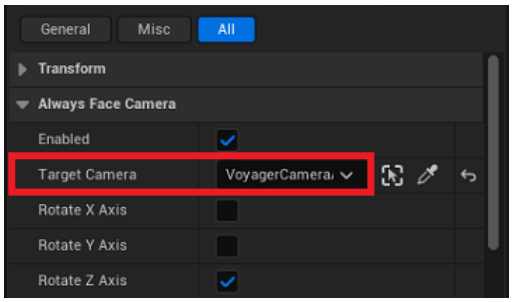
7. With the **VoyagerCompositeActor** still selected, select the **Add** button and start entering "`Always`..." to add an **Always Face Camera** component.

Voyager Composite - Always Face Camera

8. The **Always Face Camera** details are added to the **VoyagerComposite Details** tab immediately following the **Transform** section.

   If you don't see the **Always Face Camera** details, select the **All** tab.

9. In the **Always Face Camera** section, set the **Target Camera** to the corresponding **VoyagerCameraActor**.



Voyager Composite Details - Always Face Camera

10. Repeat this procedure to configure each **VoyagerCompositeActor**, if there are more than one.

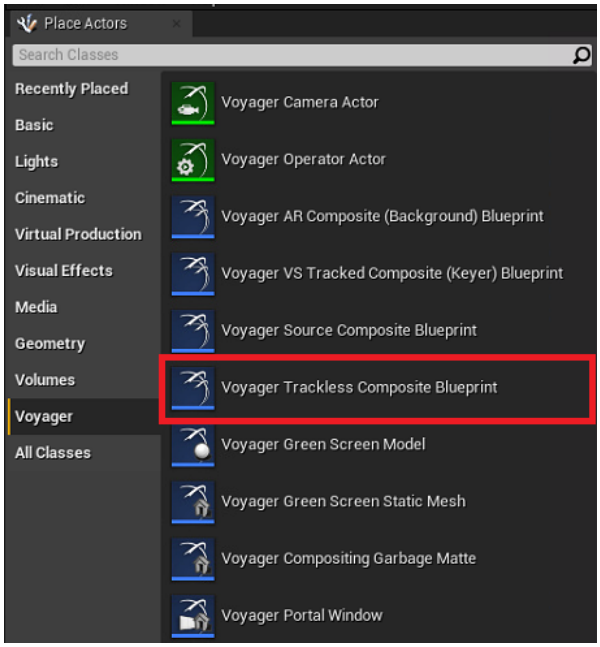
11. Select **Save**.

Method 2

In the second method, you will replace each composite actor with a **Voyager Trackless Composite** actor.

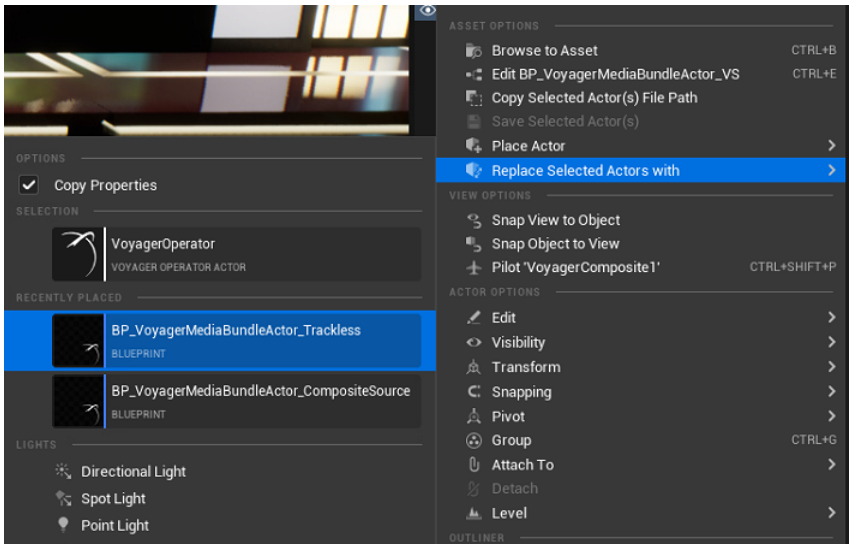
To replace a VoyagerCompositeActor:

1. In the **Place Actors** tab, from the **Voyager** category, select the **Voyager Trackless Composite Blueprint** and drag it into the level.



Place Voyager Trackless Composite Blueprint

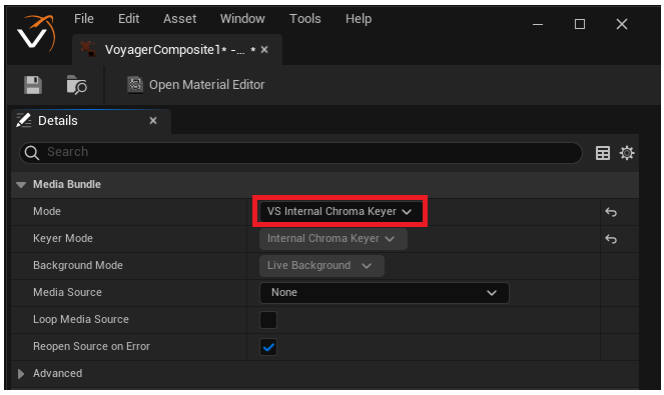
2. In the **Outliner**, select a composite.
3. Right-click and select **Replace Selected Actor with > BP_VoyagerMediaBundleActor_Trackless**.



Replace Voyager Composite Actor

4. In **Details > Media Bundle**, double-click the **Media Bundle** icon to open the **Media Bundle** editor.

5. In the editor, set the **Media Bundle Mode** to **VS Internal Chroma Keyer**, **VS External Keyer** or **VS Composure Input** — depending on your chroma keyer setup.

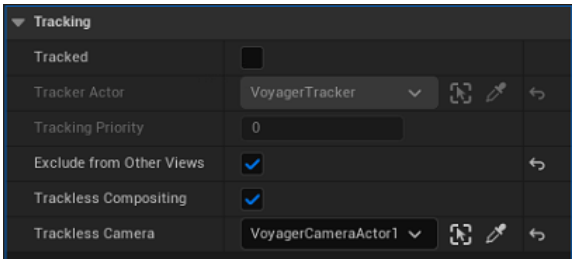


Media Bundle Details - Select Mode

6. Select **Save** and close the editor.
7. If you have more than one composite and want to make sure that the camera for this composite doesn't see the other composite, in **Details > Tracking**, do the following:

   a) Select the **Exclude from Other Views** checkbox.

   b) From the **Trackless Camera** drop-down, select the camera actor associated with the composite.



Trackless Mode - Exclude from Other Views

8. In the **Outliner**, delete the **BP_VoyagerMediaBundleActor_Trackless** actor.
9. Select **Save**.

To remove unused components:

1. In the **Outliner**, right-click the **VoyagerTracker** actor and select **Edit > Delete** to remove it from the project.
2. Similarly, remove the **BP_FreeRoaming_Cam Blueprint** from the **Outliner**, if one exists.
3. Select **Save**.

Launching a Voyager Project

Now that you've created your Voyager project, you can launch and play it to ensure it's behaving as you expect.

You can launch a Voyager project in several ways, as follows:

Locally, using Lucid Studio Renderer Service

Remotely, using Lucid Studio

From the desktop icon

Directly from a Voyager project file

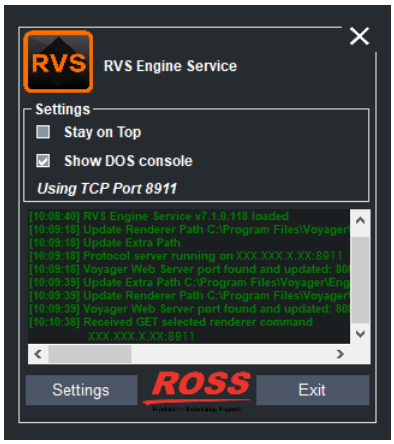
From Voyager Switchboard (Virtual LED and Virtual LED with Set Extension projects), see Using Voyager Switchboard Launcher.

For information on playing your Voyager project, see Playing a Voyager Project.

Launching a Voyager Project Locally

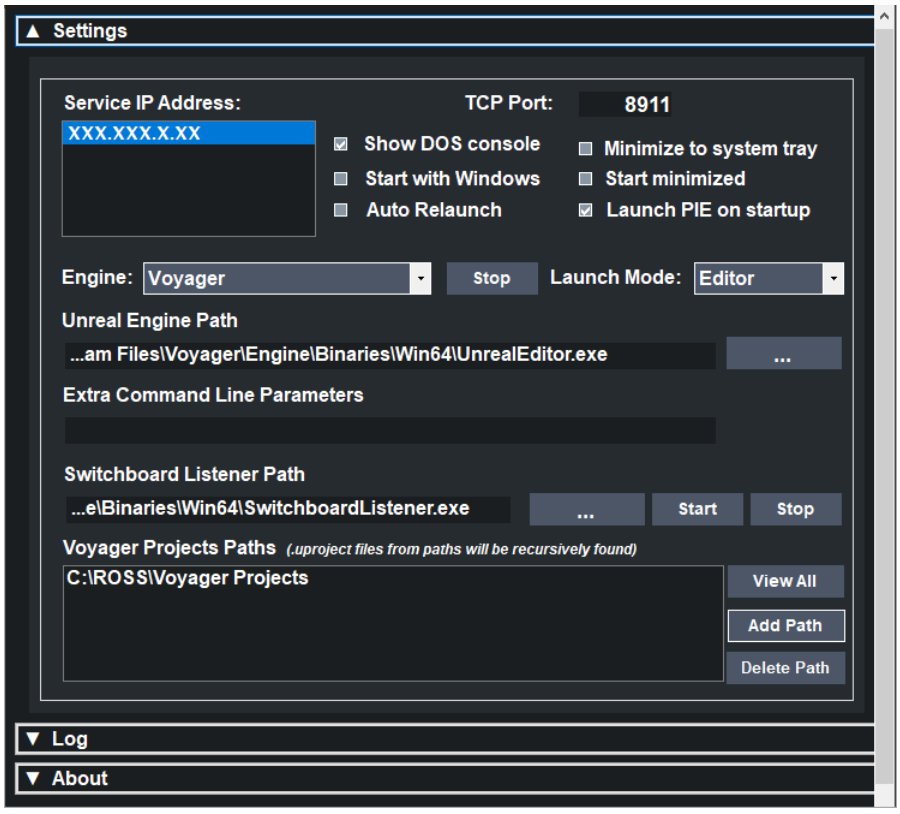
To launch your Voyager project locally:

1. Select the **RVS Engine Service** icon in the Windows tray to launch **RVS Renderer Service** if it is not already running.



RVS Engine Service

2. Then select **Settings**.



RVS Engine Service Settings

3. In **Settings**, from the **Renderer** drop-down, select **Voyager**.
4. From the **Launch Mode** drop-down, select whether to launch your project in **Editor** or **Game** mode. If **Game** is selected, this setting will over-ride the **Launch PIE on startup** setting (if it is selected).

5. Select the **Browse** button beside the **Unreal Engine Path** field and navigate to the location of your Unreal Engine executable file **(UnrealEditor.exe**), select the file and select **Open**.

   Typically, this file is located in **C:\Program Files\Voyager\Engine\Binaries\Win64**.

6. To launch Voyager in the editor window before going on air, in the **Extra Command Line Parameters** field, enter `-piewin`.

   OR

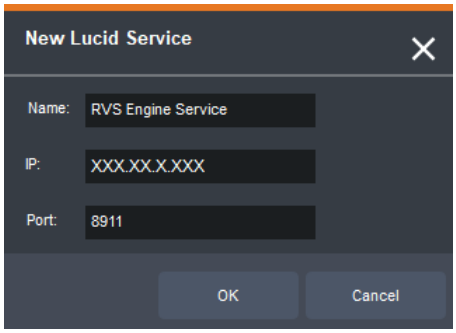
   Leave the **Extra Command Line Parameters** field blank if you want to launch Voyager while on air.

7. In the **Voyager Projects Paths** window, select the path to your Voyager projects.
8. Select **View All** to display the list of projects in that location.
9. Select the project (**name.uproject** file) you want to open and select **Run**.
10. Then select **Close**.

Launching a Voyager Project Remotely from Lucid Studio

To launch your Voyager project remotely (from Lucid Studio):

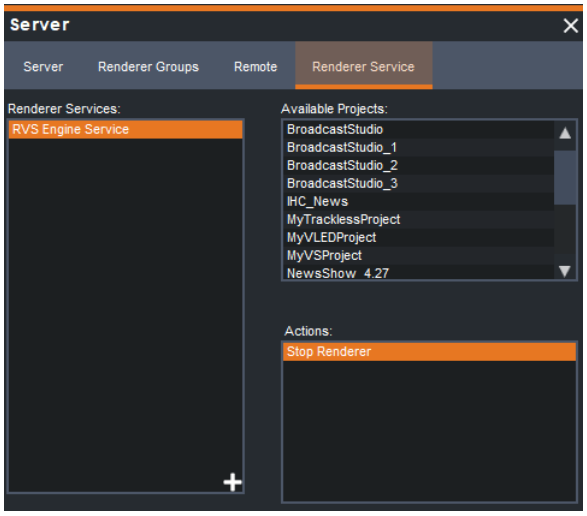
1. Configure **RVS Engine Service** as described in Steps 1 to 6 in To launch your Voyager project locally.
2. In Lucid Studio, in the **Server** panel, select the **Renderer Service** tab.
3. Select the **+** sign in the bottom-right corner of the **Renderer Services** pane to add the **RVS Engine Service** that resides on your Voyager engine to the list, if it is not already there.



Add RVS Engine Service

4. In the **Name** field, enter a unique name for the **Service**, for example, **RVS Engine Service**.
5. In the **IP** field, enter the IP address of the Voyager machine on which the **RVS Engine Service** is running.
6. In the **Port** field, enter the **TCP Port** number found in the **Settings** window of the **RVS Engine Service** on the Voyager engine.

   The default port is **8911**.
7. Then select **OK**.
8. From the **Renderer Services** list, double-click the **Lucid Service** instance you just created for Voyager, to load a list of available projects on that engine.
9. From the **Available Projects** list, double-click the project you want to open.



Select Voyager Project

Launching a Voyager Project from the Desktop Icon

To launch your Voyager project from the desktop icon:

1. Select the **Voyager** icon on your desktop.
2. From the **Recent Projects** section, select the project you want to launch and select **Open Project**.
3. If the project you want isn't displayed, select the **More** button.

Launching a Voyager Project From a Project File

To launch your project directly from a project file:

1. Navigate to the location on your PC where your project file is located.
2. Double-click the project file (**name.uproject**).

   If this is the first time you are launching this project, the **Select Unreal Engine Version** dialog opens.
3. From the drop-down, select the latest build from the drop-down and select **OK**.

   Subsequently, your project will launch immediately upon double-clicking the project file.

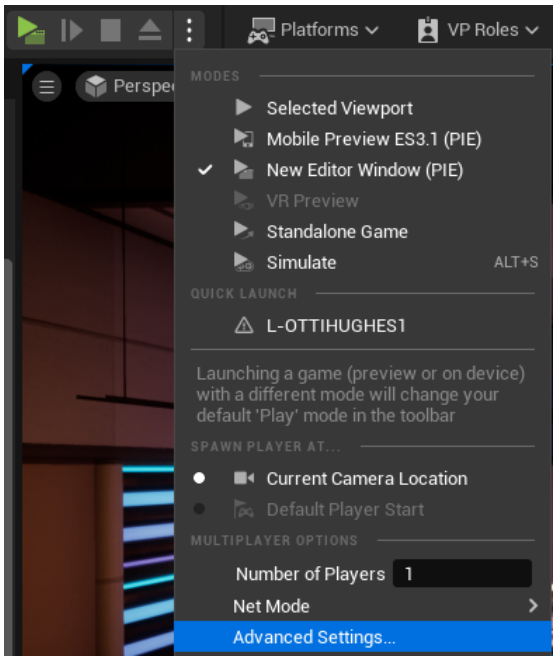
Playing a Voyager Project

With your project launched, you can now play the project in the editor and check your results.

For Virtual LED projects with multiple screens, you can play your project from the **Voyager Switchboard Launcher**. See Using Voyager Switchboard Launcher.

To play your project:

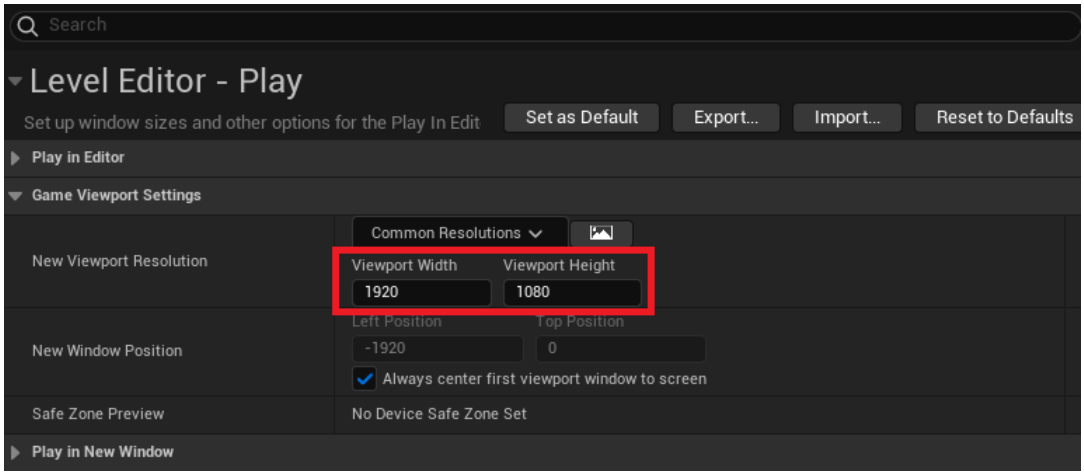
1. Launch your Voyager project.
2. In the main toolbar of Voyager, select the 3 vertical dots beside the **Play** controls, and from the context menu, select **Advanced Settings**.



Advanced Settings

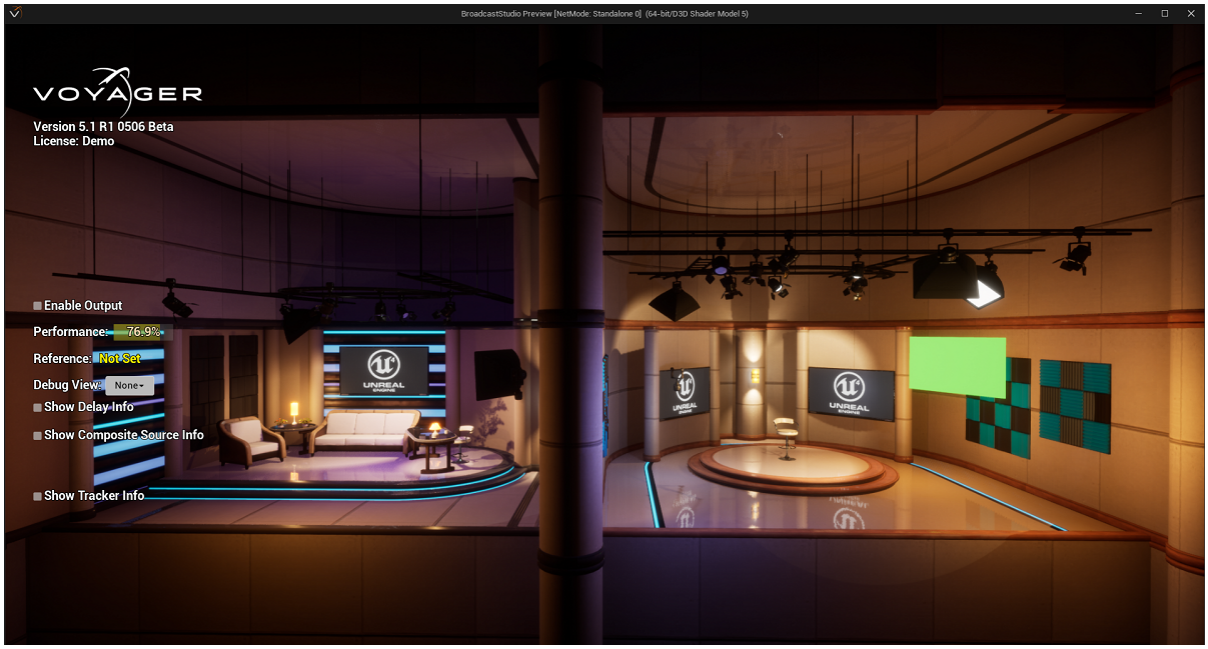
The **Editor Preferences** window opens with **Level Editor - Play** displayed.

3. In the **Game Viewport Settings** section, check that the **New Viewport Resolution** is set to **1920 x 1080** and close the **Editor Preferences** window.



New Window Size Setting

4. Now, select the arrow beside the **Play** icon and select **New Editor Window (PIE)**.



Project Playout

The white text you see is informational and is not visible on air.

The following information is provided if selected:

- **Enable Output**: Puts the signal out for broadcast (selected by default).
- **Performance**: Indicates the level of performance.
- **Reference**: Indicates whether the reference signal is locked or unlocked.
- **Debug View**: Provides a selection of different views for information or troubleshooting.
- **None**: Selected by default.

  ★Do not select other views while on air, as this could interfere with the output.

- **Show Delay Info**: Shows the breakdown of frame delays for tracking, composite input and set extension (if applicable). The set extension delay can be changed in this screen, if necessary.
- **Show Composite Source Info**: Shows information about the incoming VoyagerComposite video feed (the talent feed inside the set).
- **Show Tracker Info**: Displays camera tracking data and incoming camera position data. Red **X's** indicate a problem.

Publishing and Deploying a Voyager Project

Starting with Voyager 8.0, you can now publish Voyager projects to the XPression Project Server and then deploy those projects to other Voyager machines using the RVS Project Manager application. For information on using the RVS Project Manager, see the *RVS Project Manager User Guide*, included with your documentation.

For information about setting up shows and categories in the XPression Project Server Manager, see the *XPression Project Server User Guide*.

The RVS Project Manager application can be accessed from Voyager using the icon  at the top-right corner of the user interface, or from the RVS Project Manager desktop icon.

Chroma Keying

Once you've created a project from a template or made an existing project compatible with Voyager, you may need to make some adjustments in the chroma keying.

There are a couple of ways to do this.

Composure Chroma Keying

Color Difference Chroma Keying

You might find it helpful to first open the **Voyager Editor Preview Output** window. This allows you to see how the set and talent will look in **Play** mode without actually playing it.

To open the Voyager Editor Preview Output window:

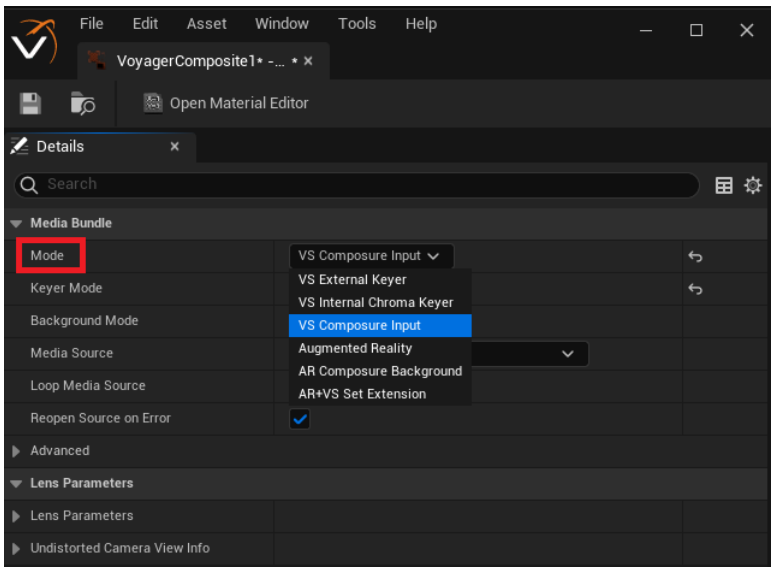
1. In the **Outliner**, select the **VoyagerOperator** actor.
2. In the **Details** tab, scroll down to and expand the **Editor Preview Output** section.
3. Select the **Enable Editor Output Capture** checkbox.

Composure Chroma Keying

This method uses the **VS Composure Input** mode and **Multi Pass Chroma Keying**. You will also need to create a composure shot and add a media plate, as described below.

To select the chroma keying mode:

1. In the **Outliner**, select **VoyagerComposite1**.
2. In the **Details** panel, scroll down to the **Media Bundle** section and double-click the **VoyagerComposite1 Media Bundle** icon to open the **Details** editor.
3. In the **Details** tab, in the **Media Bundle** section, from the **Mode** drop-down, select **VS Composure Input**.

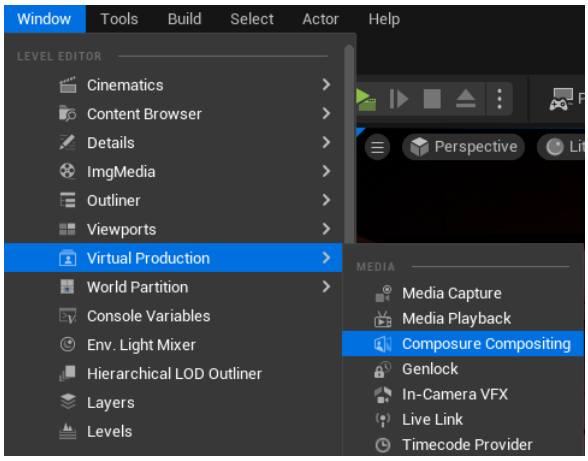


Select VS Composure Input

4. Select **Save** and close the editor.

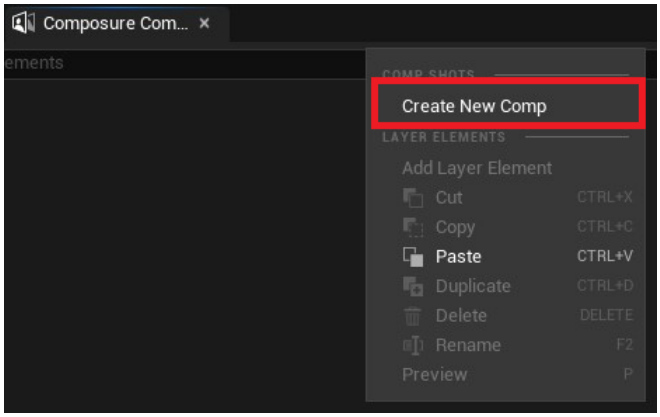
To create a composure shot:

1. Select **Window > Virtual Production > Composure Compositing** and dock the panel that opens to the right of the **Content Browser**.



Composure Compositing

2. In the **Composure Compositing** panel, right-click and select **Create New Comp**.

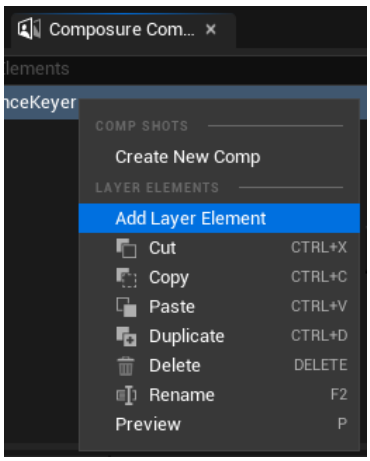


Create New Comp

3. In the **Pick an Comp Class** window, select **Empty Comp Shot**.
4. In the **Composure Compositing** panel, name the comp shot (e.g., **ColorDifferenceKeyer**).

To add a media plate element:

1. In the **Composure Compositing** panel, right-click on the new comp shot and select **Add Layer Element**.



Add Layer Element

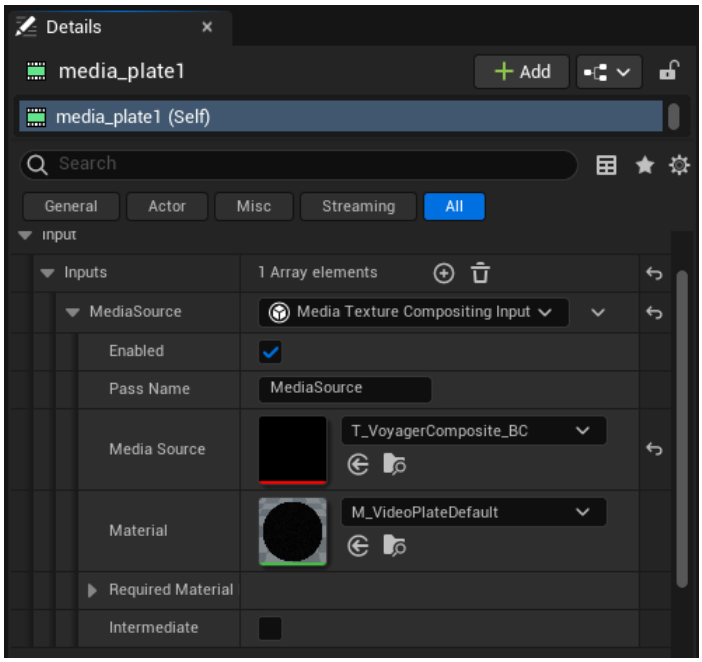
2. In the **Pick an Element Type** window, select **Media Plate**.

   The **Media Plate** element is nested inside the **ColorDifferenceKeyer** comp shot and by default is named **media_plate1**.

3. In the **Outliner**, move the **ColorDifferenceKeyer** and **media_plate1** element into the **Voyager** folder to be able to find them easily.
4. In the **Outliner**, select **VoyagerComposite1**.
5. In the **Details** panel, in the **Composure** section, from the **Composure Input Pass** drop-down, select **media_plate1**.

6. In the **Outliner**, select **media_plate1** and in the **Details** panel, scroll down to the **Input** section.

   You'll see that the appropriate selections have been made by default, as shown in the image below:



Media_plate1 Details

7. Continue with **Multi Pass Chroma Keying**.

Multi Pass Chroma Keying

The settings and parameters for **Multi Pass** chroma keying are found in the **media_plate1** actor. You will need to select the **Multi Pass** chroma keyer, then select two colors to be keyed out and make some adjustments in the material parameters to get a sharp keyed image.

You may find it helpful to zoom your camera in on the talent while adjusting the **Chroma Keyer** parameters. This will make it easier to see the changes.

To select the Multi Pass Chroma Keyer:

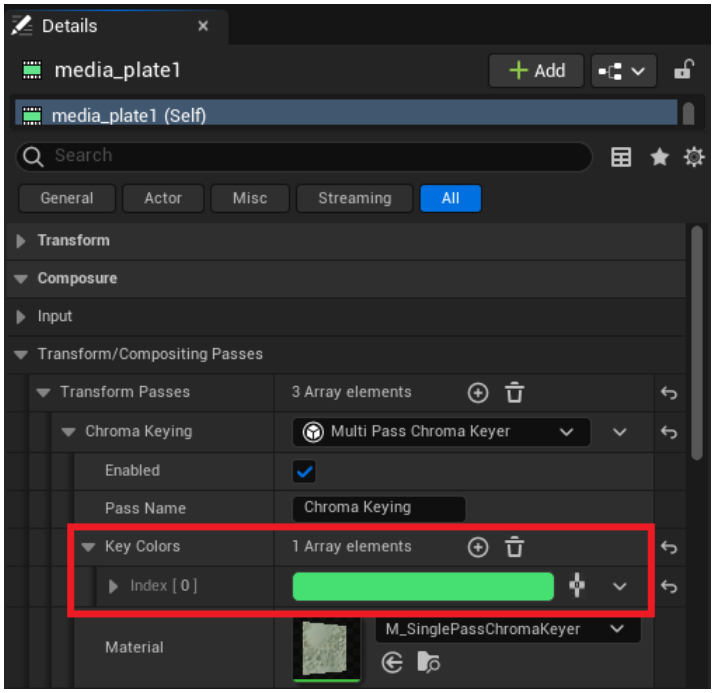
1. In the **Outliner**, select **media_plate1**, if it is not already selected.
2. In the **Details** panel, in the **Transform/Compositing Passes** section, expand **Transform Passes** and make sure that **Multi Pass Chroma Keyer** is selected in the **Chroma Keying** drop-down.

To select colors to be keyed out:

1. In the **Content Browser**, open the **Voyager > LiveSources > VoyagerComposite_InnerAssets** folder and double-click the **T_VoyagerComposite_BC** media texture to open its editor.

   You may need to re-size the editor, so that you can see the editor and the **Outliner** and **Details** panels.
2. With **media_plate1** selected in the **Outliner**, expand **Chroma Keying** and from the **Material** drop-down, select **M_SinglePassChromaKeyer**.
3. Select the **+** icon beside **Key Colors** to add an **Array element**.

4. Select the checkered bar to open the **Color Picker** and select the eye-dropper.
5. Use the eye-dropper to select the main color in the background of the **T_VoyagerComposite_BC** media texture.
6. Drag and drop the selected color to the color bar at the top of the **Color Picker**.



Select Main Key Color (Multi Pass)

7. Select the **+** icon beside **Key Colors** to add a second **Array element**.
8. Select the second checkered bar to open the **Color Picker** and select on the eye-dropper.
9. Use the eye-dropper to select the second most prominent color in the background of the **T_VoyagerComposite_BC** media texture.
10. Close the **T_VoyagerComposite_BC** media texture.

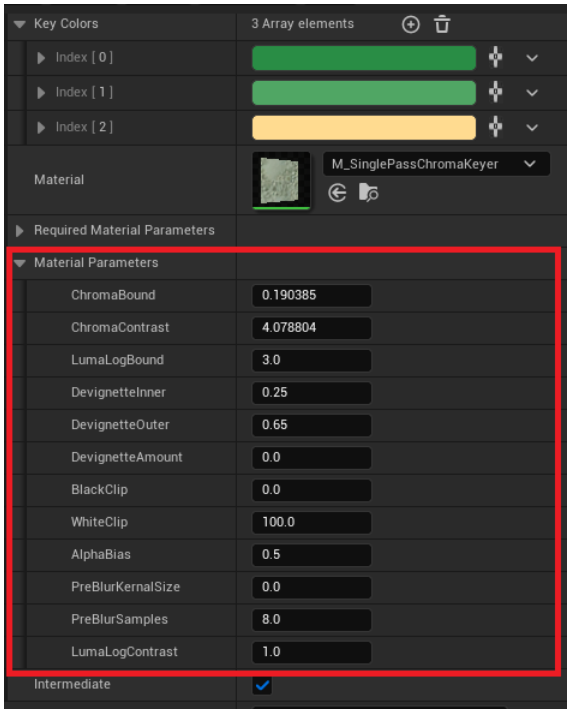
To select the Despill color:

1. In the **Details** tab, scroll down to **Despill** and expand that section.
2. Select the **+** icon beside **Key Colors** to add an **Array element**.
3. Select the checkered bar in the **Despill Key Colors** and select the color saved to the color bar at the top of the **Color Picker**.

   This helps to remove any reflection spilling onto the talent from the background.
4. Select **OK** to close the color picker.

To adjust the material parameters:

1. Zoom in on the talent to be able to see details like the hair better.
2. Still in the **media_plate1 Details** panel, in the **Chroma Keying** section, expand the **Material Parameters** section.



Material Parameters - Composure Chroma Keying

3. Adjust the **Material Parameters** as necessary to achieve a good image.
4. If further adjustment is necessary, increase or decrease the other parameter values one at a time and check the result to see if the image is improved.

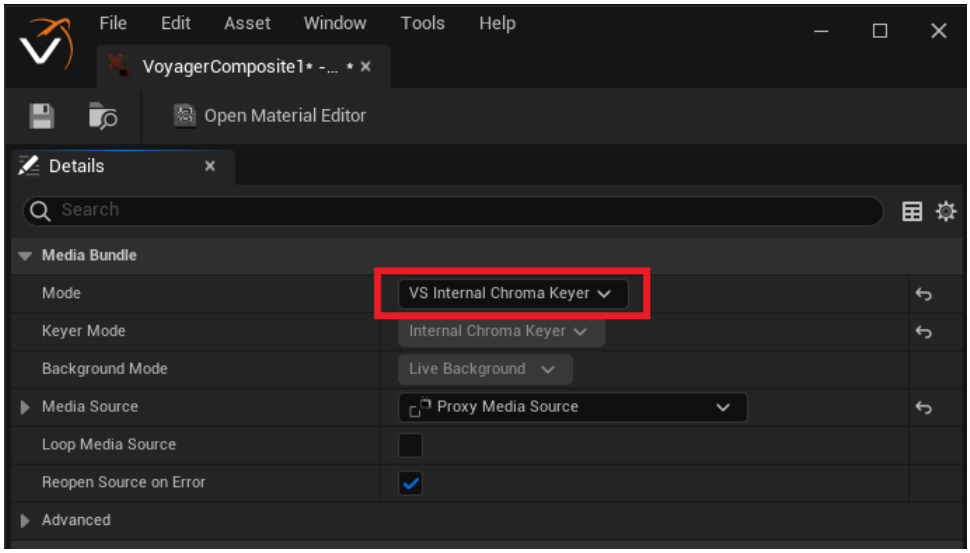
   You can return to the default values at any time by selecting the yellow reset arrow that appears beside the parameter field when it's changed.

Color Difference Chroma Keying

This method uses the **VS Internal Chroma Keyer** mode and **Single Pass Chroma Keying**.

To select the chroma keying mode:

1. Navigate to the **Voyager > LiveSources** folder and double-click the **VoyagerComposite1 Media Bundle** to open the **Details** tab.
2. In the **Details** tab, in the **Media Bundle** section, from the **Mode** drop-down, select **VS Internal Chroma Keyer**.



Select VS Internal Chroma Keyer

3. Select **Save** and close the editor.
4. Continue with **Single Pass Chroma Keying**.

Single Pass Chroma Keying

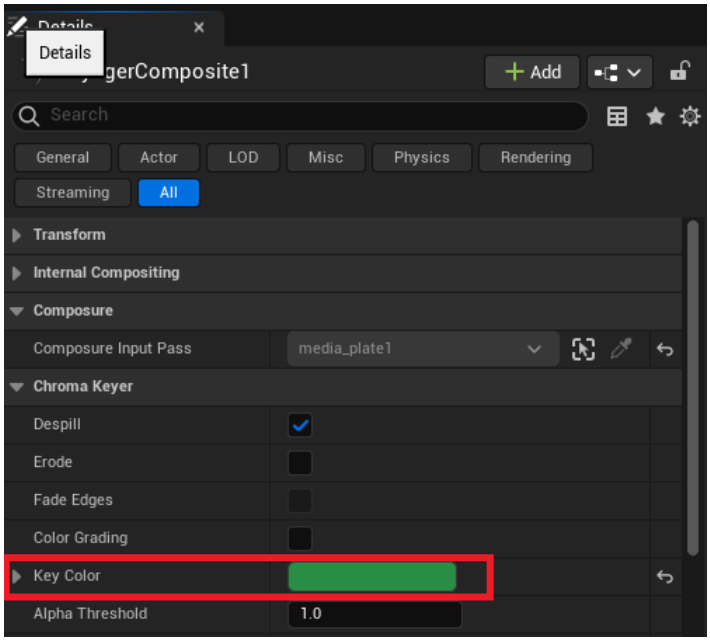
The settings and parameters for **Single Pass** chroma keying are found in the **VoyagerComposite1 Media Bundle**. You'll need to select the color to be keyed out and make some adjustments in the material parameters to get a sharp keyed image.

You may find it helpful to zoom your camera in on the talent while adjusting the **Chroma Keyer** parameters. This will make it easier to see the changes.

To select the color to be keyed out:

1. In the Content **Browser**, open the **Voyager > LiveSources > VoyagerComposite_InnerAssets** folder and double-click the **T_VoyagerComposite_BC** media texture to open its editor.
2. In the **Outliner**, select the **VoyagerComposite1** actor.
3. In the **Details** tab, scroll down and expand the **Chroma Keyer** section.
4. Select the **Key Color** bar to open the **Color Picker** and select on the eye-dropper.

5. Use the eye-dropper to select the main color in the background of the **T_VoyagerComposite_BC** media texture.



Select Key Color (Single Pass)

6. Close the **T_VoyagerComposite_BC** media texture.

To adjust the material parameters:

1. Zoom in on the talent to be able to see details like the hair better.
2. Still in the **VoyagerComposite 1 Details** tab, in the **Chroma Keyer** section, adjust the **Key Color** parameters as necessary to achieve a good image.

Parameter	Description
Alpha Threshold	Removes most of the core green signal while keeping detail (ie hair).
Alpha Offset	Adding a positive value brings back the core fill if the talent or object starts to become transparent. Turning on **SHOW ALPHA** in the **Debug** tab will help you make better adjustments.
Red Weight	Helps add or remove red tint from the green spill.
Blue Weight	Helps add or remove blue tint from the green spill.
Clip Black	Adds or removes blacks from the signal.
Clip White	Adds or removes whites from the signal.
Despill	• **Despill Hue Range** Desaturates or saturates the signal to neutralize the hue. • **Despill Amount** Replaces the despill with a color range depending on the value input.

3. Begin by adjusting the **Alpha Threshold** and **Alpha Offset** parameters slightly to remove any transparent areas and sharpen the image.

Make the minimum amount of adjustment possible to get a good image.

4. If further adjustment is necessary, increase or decrease the other parameter values one at a time and check the result to see if the image is improved.

   You can return to the default values at any time by selecting the yellow reset arrow that appears beside the parameter field when it's changed.

Switching Cameras in a Multi-Camera Setup

You can switch between multiple cameras in a set to show the same scene from a different perspective. This requires defining a number of virtual cameras to be associated with the physical cameras in your set and adding nodes to the level blueprint that will set the active tracker actor.

Follow the steps below to set up camera switching:

Adding a Voyager Operator to your Project (if there isn't one already)

Adding Voyager Trackers

Adding Voyager Tracker Actors

Adding Camera-Switching to the Level Blueprint

Adding a Voyager Operator to your Project

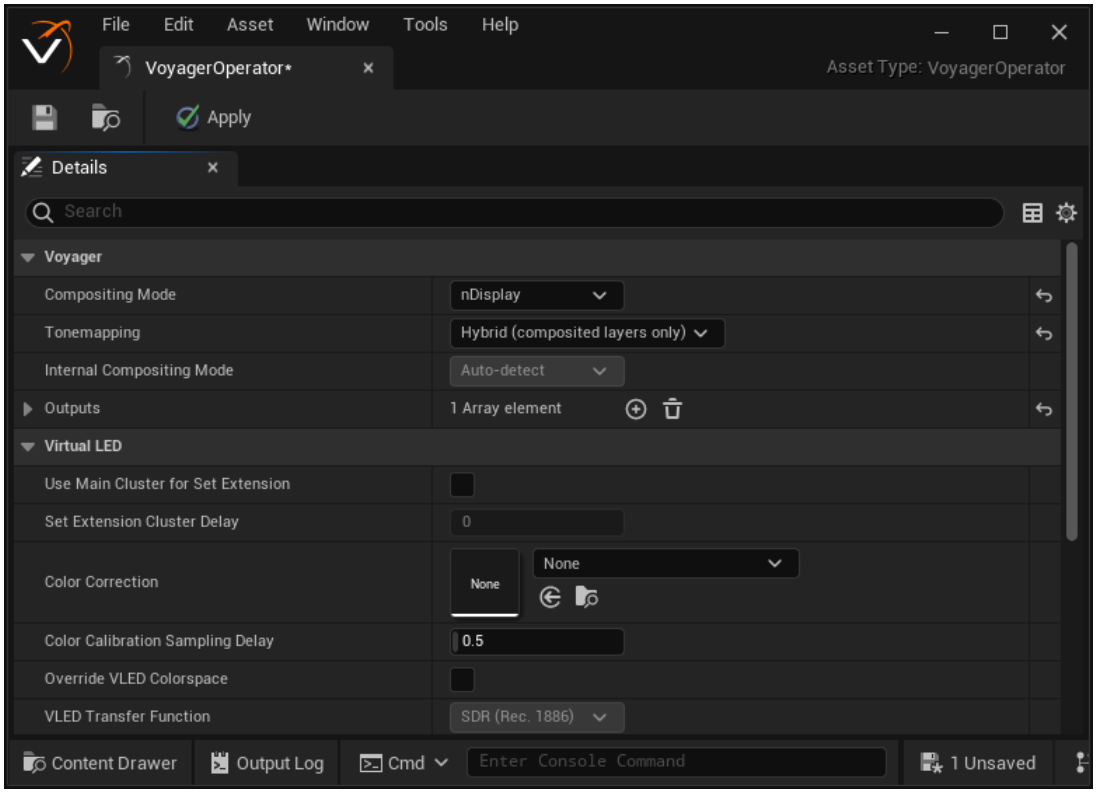
If you have used the Virtual LED template to create your project, you will need to add a **Voyager Operator** to your project in order to do camera-switching. All other templates already have a **Voyager Operator**.

If you already have a **Voyager Operator** in your project you can skip these instructions and continue with Adding Voyager Trackers.

To add a Voyager Operator to your project:

1. In the main toolbar, from the **Voyager** drop-down, select **New Voyager Operator**.
2. In the **Save Asset As** window, navigate to the **Voyager** folder and add a new folder named **Operator**.
3. Then save the new **Voyager Operator** in the **Operator** folder, with a name like **VoyagerOperator**.

   The **VoyagerOperator** editor opens and the **VoyagerOperator_InnerAssets** folder is automatically created.
4. In the **VoyagerOperator** editor, in the **Voyager** section, from the **Compositing Mode** drop-down, select **nDisplay**.



Voyager Operator Editor - nDisplay

5. From the **Tonemapping** drop-down, select either the **Native** or **Hybrid** option.
   - **Native** - The Unreal Engine native tonemapping and post-processing will be applied to everything in the level, including the incoming camera feeds. This setting is not recommended in internal compositing as the look of the camera feeds will be affected.
   - **Hybrid** - A broadcast tonemapping (designed to preserve the look of the incoming camera feed) will be applied only to the incoming camera feeds. Native tonemapping and post-processing will only affect the graphics and not the incoming camera feeds used for the composite layers.
6. From the **Internal Compositing Mode** drop-down, select from the following options:

Auto-detect — uses Linear compositing (engine default) unless the Voyager Media Bundle on the first compositing layer is set to **External Keying** and **Tonemapping** is set to **Hybrid**.

Linear — recommended for multiple compositing layers, as it supports the most features and layering combinations.

Non-Linear — recommended for a single layer using external keying, as it supports fewer layering and post-processing combinations but replicates the external compositing typical of broadcast devices.

★In **Non-Linear** mode, pixels in the virtual graphics that are hidden by composite objects will still generate bloom, as if they were visible. This mode is intended for use with external keyers. It could have undesired effects on the edges of internal chroma keyers.

★Additional live sources that are fill-only without using chroma keying can be used in either mode, based on the nature of the main composite's keying method.

7. Select **Save** and close the editor.
8. In the **Content Browser**, open the **Voyager > Operator** folder and then select and drag the **VoyagerOperator** into the level.

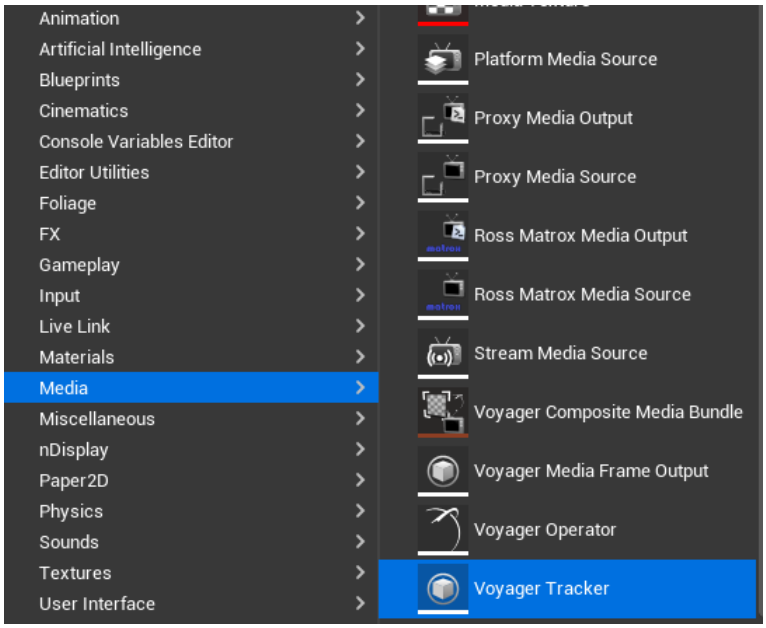
   The location of the **VoyagerOperator** actor in the scene is not important. It can be placed anywhere.

Adding Voyager Trackers

Typically, your project will have at least one **Voyager Tracker** (except for Voyager Trackless projects). For multi-camera switching, you'll need to add an additional **Voyager Tracker** for each physical camera in your set to which you want to be able to switch.

To add Voyager Trackers:

1. In the **Content Drawer**, in the **Voyager > Tracker** folder, right-click in the empty space and select **Media > Voyager Tracker**.



Add Voyager Tracker

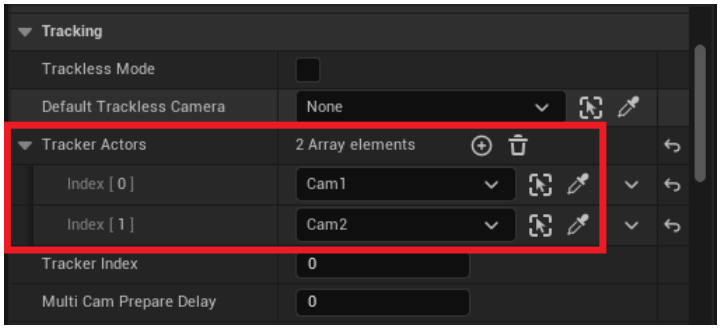
2. Rename the **Voyager Tracker** to identify the physical camera it represents (Cam1, Cam2, etc.).
3. Double-click the new **Voyager Tracker** to open the editor.
4. In the **UDP Port** field, enter a different port number from any other tracker in the scene.
5. By default, the **UDP Port** is **8456**, so you can enter **8457** for the second tracker, **8458** for the third tracker, etc.
6. Select **Save** and close the editor.
7. Drag the new **Voyager Tracker** into the scene.
8. Repeat the steps 1 to 7 for each camera view you want to display.

Adding Voyager Tracker Actors

Once you've added the required number of **Voyager Trackers** to your project, you'll need to connect them to the **Voyager Operator** by adding **Voyager Tracker Actors**..

To connect Voyager Tracker Actors:

1. In the **Outliner** tab, select **VoyagerOperator**.
2. In the **Details** tab, in the **Tracking** section, select the **+** icon to add a **Tracker Actor Array** element for each tracker you added to the scene.
3. From the drop-down for the first new element (**Tracker Actor Index [0]**), select your first **VoyagerTracker** (e.g., Cam1).



Add Tracker Actors

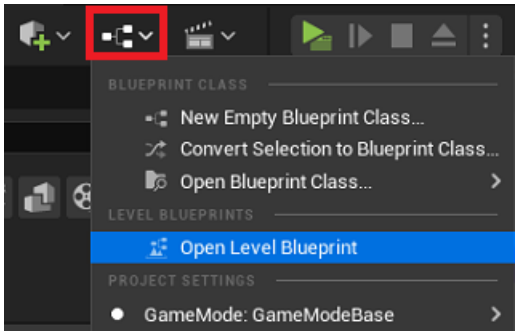
4. From the drop-down for the second new element (**Tracker Actor Index [1]**), select your second **VoyagerTracker** (e.g., Cam2).
5. Repeat steps 3 and 4 for each additional tracker.

Adding Virtual Camera-Switching to the Level Blueprint

Now you'll need to add some nodes to the level blueprint to switch cameras. The trigger to switch cameras can come from a number of inputs, such as a switcher node, a **Lucid_Exec** node, a **RossTalk** node, or a **Voyager Timecode Event** node, depending on your workflow.

To open the level blueprint:

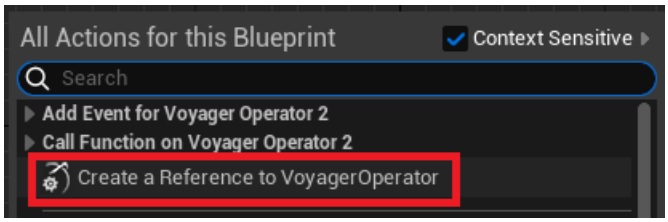
1. In the **Outliner**, select the **VoyagerOperator**.
2. Then select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.



Open Level Blueprint

To create the first camera switch graph:

1. In the **Event Graph**, right-click in an empty section of the graph and select **Create a Reference to VoyagerOperator**.



Create a Reference to VoyagerOperator

2. Right-click again and in the **All Actions for this Blueprint** list, in the **Search** field, start typing the first part of the input node name and then select the node from the list:

   For example:

   Type `RossTalk` to add an **On RossTalk GPI Event** node.

   OR

   Type `Lucid` to add a **Lucid Exec** node.

   OR

   Type `Timecode` to add a **Voyager Timecode Event** node. For further information see how to execute a camera switch using the On Voyager Event node

3. Select the **On GPI** pin (or **Exec 1** on a Lucid input node, or the **Exec** output pin on a **Voyager Timecode Event** node) and drag out a connection to place a new node.

4. In the **Executable** actions list, in the **Search** field, start typing `Set Active Tracker Index` and add a **Set Active Tracker Index** node to the input node.

   If your search doesn't come up with the **Set Active Tracker Index** node, clear the **Context Sensitive** checkbox and try again.

5. In the input node, set the applicable parameters as described below:

   - In a **Lucid Exec** node, in the **Var Name** field, enter a name (e.g., Camera_Switch_1).
   - In a **RossTalk** node,
     - set the **Condition to Match Number** only.
     - Set the **GPI As Number** field to the **Index** number of the tracker actor to which it refers (e.g., "0" for Cam1, "1" for Cam2, etc.) as defined in the **VoyagerOperator**.
   - In a **Voyager Timecode Event** node, set the **Timecode Event Action** value to the number of the tracker actor to which it refers (e.g., "0" for Cam1, "1" for Cam2, etc.) as defined in the **VoyagerOperator**.

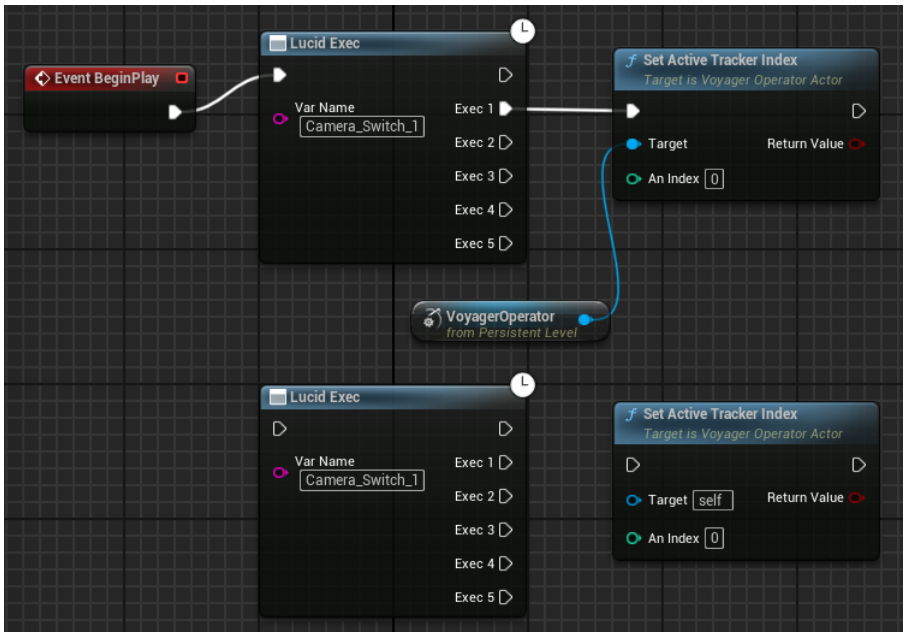
6. In the **Set Active Tracker Index** node:

   - Connect the **Output** pin of the **VoyagerOperator** reference node to the **Target** pin.
   - Set the **An Index** field to the **Index** number of the tracker actor to which it refers ("0" for Cam1, "1" for Cam2, etc.) as defined in the **VoyagerOperator**.

7. Add the **Event Begin Play** node to the input node.
8. Select **Save** and then **Compile**.

To add additional camera switches:

1. Left-click and drag around the input node and the **Set Active Tracker Index** node created above to select them, and then right-click and select **Duplicate** to create another set of identical nodes.



Duplicate Camera Switching Graph

2. In the second input node, change the applicable parameter as follows:

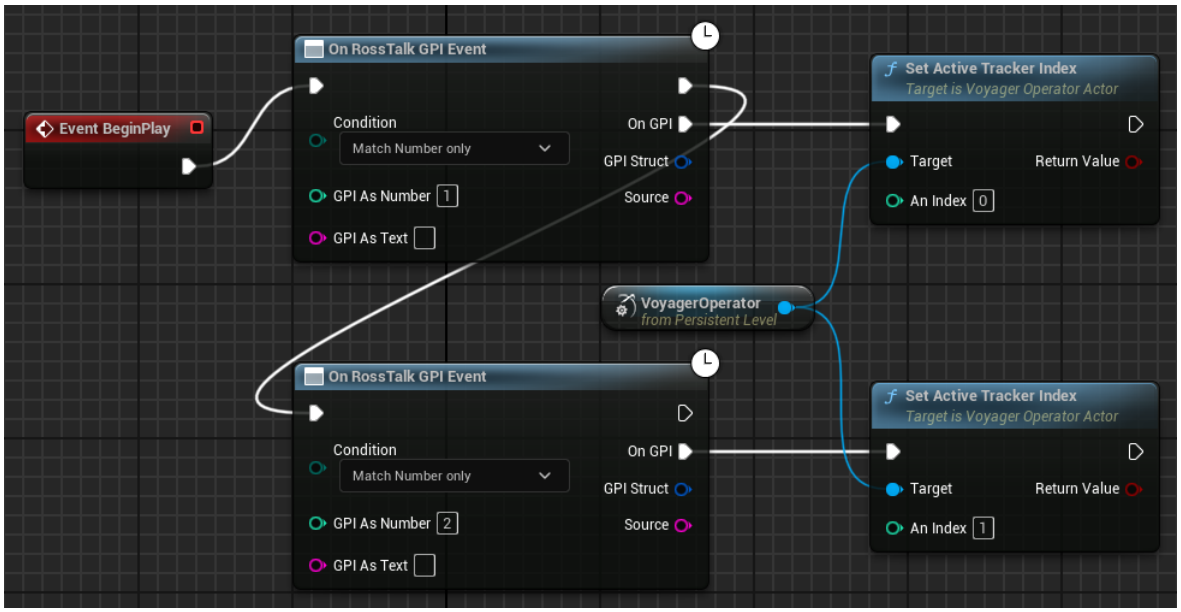
   - In a second **RossTalk** node, set the **GPI As Number** value in the new node to correspond to the **Index** number of the second tracker actor defined in the **VoyagerOperator**.
   - In a second **Lucid Exec** node, change the **Var Name** to correspond to the **Index** number of the second tracker actor defined in the **VoyagerOperator**, which would be called in a **Renderer Logic** function in Lucid Studio (Method 1).

     Alternatively, you can have one **Lucid Exec** node that switches up to 5 cameras and connect the **Exec 2** pin to the second camera, the **Exec 3** pin to the third camera, etc. (Method 2).

     If you need more than 5 camera switches, you'll need to use Method 1.
   - In a second **Voyager Timecode Event** node, set the **Timecode Event Action** value to correspond to the number of the 2nd tracker actor defined in the **VoyagerOperator**.

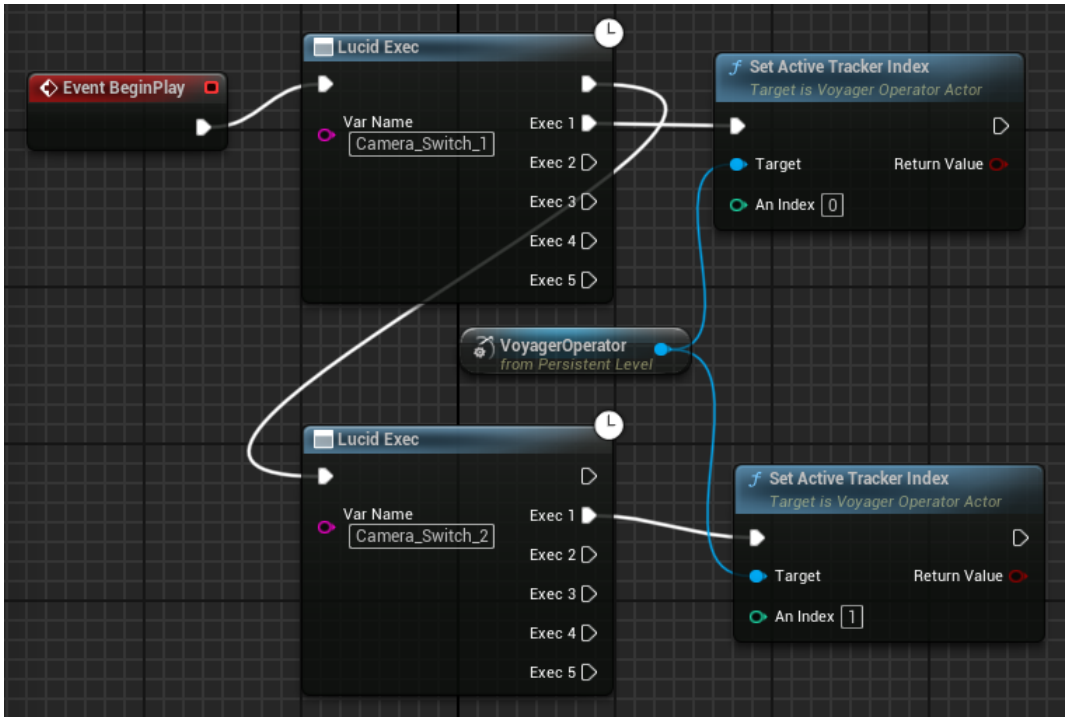
3. In the second **Set Active Tracker Index** node, set the **An Index** value to the **Index** number of the second tracker actor.
4. Repeat steps 1 to 3 for any additional physical cameras to which you want to be able to switch.
5. Connect the **Exec** output pin of the first input node to the **Exec** input pin of the second input node and connect the **Exec** output pin of the second input node to the **Exec** input pin of the third input node, as is necessary for the number of input nodes.
6. Select **Save** and then **Compile**.

   A blueprint for two cameras with a RossTalk trigger would look like the blueprint below:

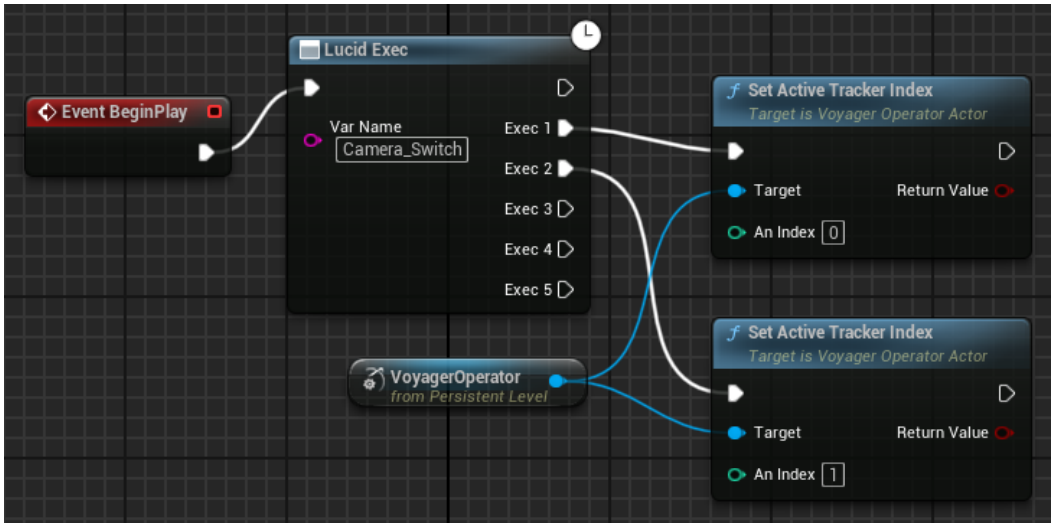


Camera Switching via RossTalk GPI

A blueprint for 2 cameras with a Lucid trigger could look like either of the blueprints below:



Camera Switching via Lucid Exec (Method 1)



Camera Switching via Lucid Exec (Method 2)

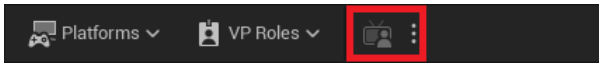
Embedded Audio Output

Embedded audio output is a feature that is available on Voyager engines that have a Matrox video card.

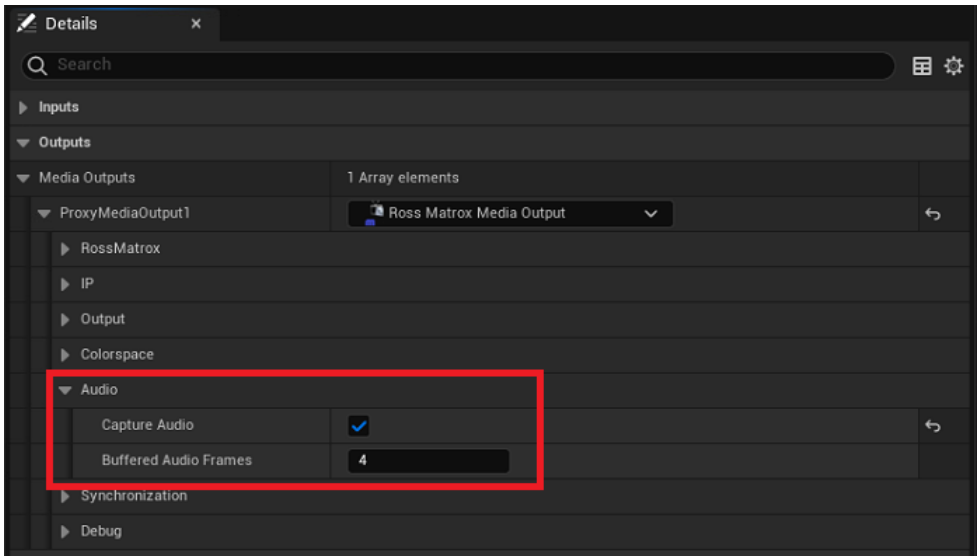
It is not currently supported by AJA.

To use embedded audio output:

1. In the main tool bar, double-click the **Media Profile** icon to open the editor.



2. In the **Outputs > Media Outputs** section, expand **ProxyMediaOutput1** and then expand **Audio**.



Embedded Audio Settings

3. Select the **Capture Audio** checkbox to capture audio from the media port.
4. In the **Buffered Audio Frames** field, enter the maximum number of frames of audio data to be stored in memory at any given time.

   The default value is 4. If you notice jumps or hitches in the input video, you can try raising this value.

Using Remote Assets

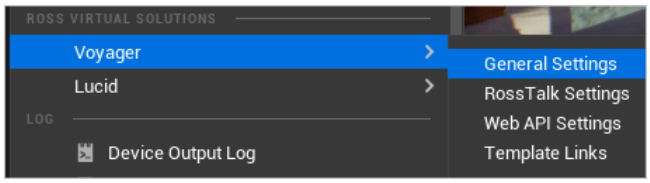
In the Voyager level blueprint, you can add nodes to apply remote assets (images or videos) to an actor or object in your scene or to multiple actors or objects. You only need to identify the URL of the remote location of the asset and specify to which actor(s)/object(s) you want to apply the asset.

The asset can then be applied using a key press, Lucid Exec, RossTalk GPI or other input node.

To be able to use remote assets, you first need to configure a few general settings.

To configure remote asset settings:

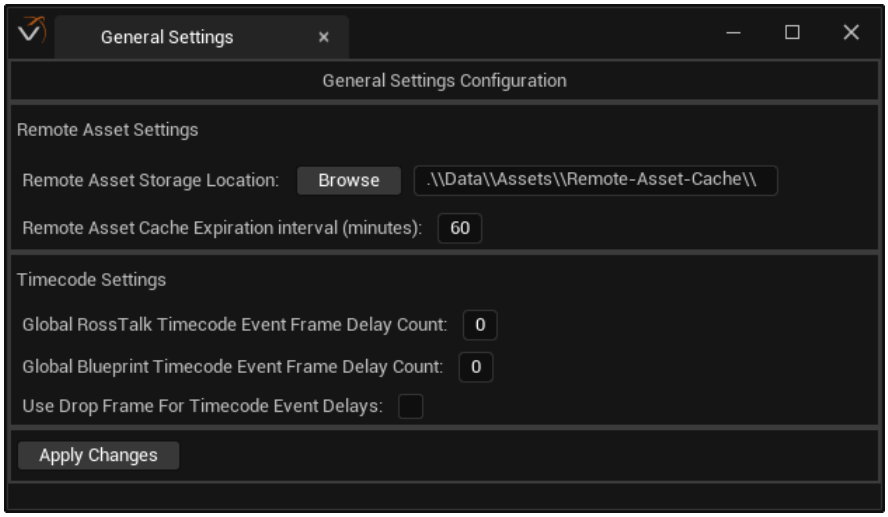
1. From the main menu, select **Window > Voyager > General Settings**.



Configure General Settings

2. In the **General Settings** tab, select the **Remote Asset Storage Location Browse** button and navigate to the folder to which you want to download your remote assets.
3. In the **Remote Asset Cache Expiration Interval** field, set the amount of time in minutes for which the locally cached assets will be used.

   After the interval, the next time the asset is requested, the system will check the remote source for updates to the asset and download any that have changed. If the remote asset can't be found or there haven't been any changes, the cached asset will be used.



General Settings - Remote Assets

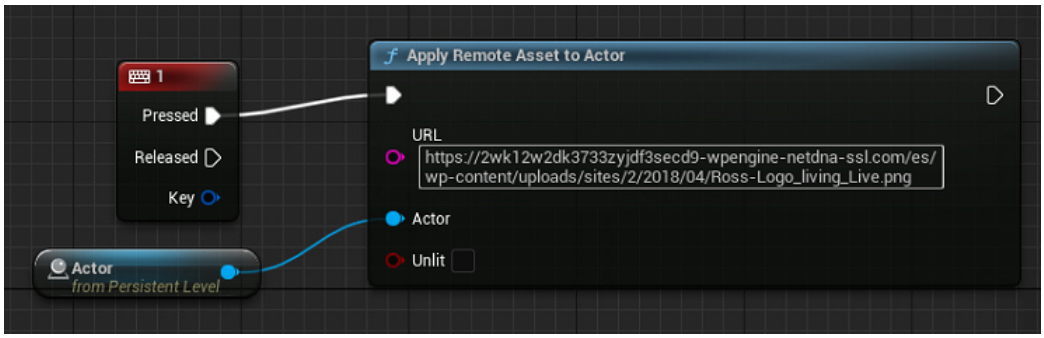
4. Select **Apply changes and save settings**.
5. Close the **General Settings** tab.

To apply a remote asset to an actor or object:

1. In the main toolbar, select **Blueprints > Open Level Blueprint**.
2. Right-click in an empty part of the **Event** graph and start typing `Apply Remote Asset to Actor`.

3. From the results, select the **Apply Remote Asset to Actor** node.
4. From the **Outliner**, drag the actor or object to which you want to apply the asset into the blueprint and connect the **Output** pin on the node that is created to the **Actor** pin on the **Apply Remote Asset to Actor** node.
5. Then drag the **Input** pin of the **Apply Remote Asset to Actor** node out and start typing `Input 1` to place an input event node.

   Alternatively, you can add a **Lucid Exec** node or a **RossTalk GPI** node or other input node.



Apply a Remote Asset to Actor

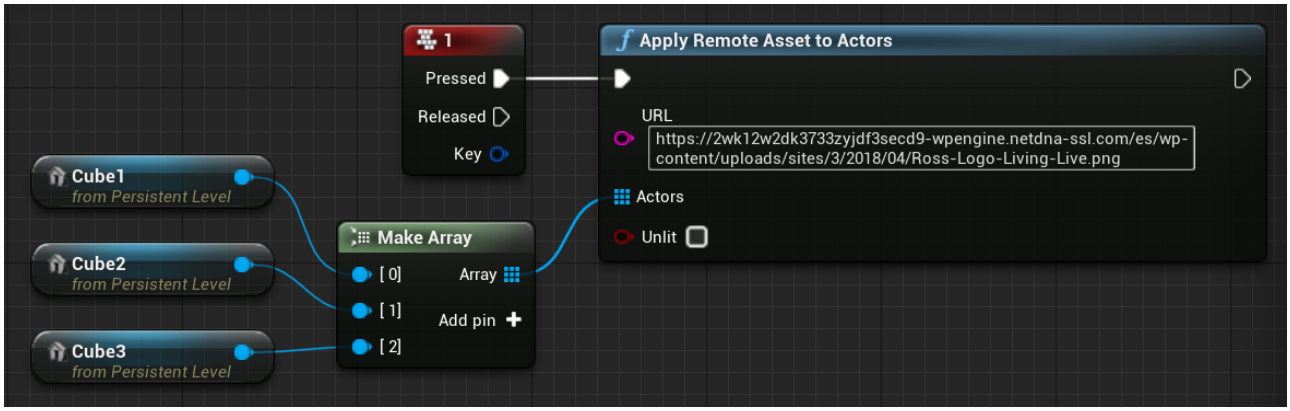
6. From the **Keyboard Events** list, select **1**.
7. In the **Apply Remote Asset to Actor** node, in the **URL** field, enter the full URL of the asset you want to apply.
8. Select the **Unlit** checkbox if you don't want lighting applied to the asset or leave it unchecked if you do want lighting applied.
9. Select **Save** and close the blueprint.

To apply a remote asset to multiple actors or objects:

1. In the main toolbar, select **Blueprints > Open Level Blueprint**.
2. Right-click in an empty part of the **Event** graph and start typing `Apply Remote Asset to Actors`.
3. From the results, select the **Apply Remote Asset to Actors** node.
4. From the **Outliner**, drag the actors or objects to which you want to apply the asset into the blueprint.
5. Right-click in the graph and add a **Make Array** node.
6. In the **Make Array** node, do the following:
   - Select the **Add** pin icon to add a pin for each actor or object node.
   - Connect the **Output** pin of each actor/object node to an **Input** pin on the **Make Array** node.
   - Connect the **Array** pin of the **Make Array** node to the **Actors** pin of the **Apply Remote Asset to Actors** node.

7. Then drag the **Input** pin of the **Apply Remote Asset to Actor** node out and start typing `Input 1` to place an input event node.

   Alternatively, you can add a Lucid Exec node or a RossTalk GPI node or other input node.



Apply Remote Asset to Multiple Actors

8. In the **Apply Remote Asset to Actors** node, in the **URL** field, enter the full URL of the asset you want to apply.
9. Select the **Unlit** checkbox if you don't want lighting applied to the asset or leave it unchecked if you do want lighting applied.
10. Select **Save** and close the blueprint.

Creating a Portal Effect

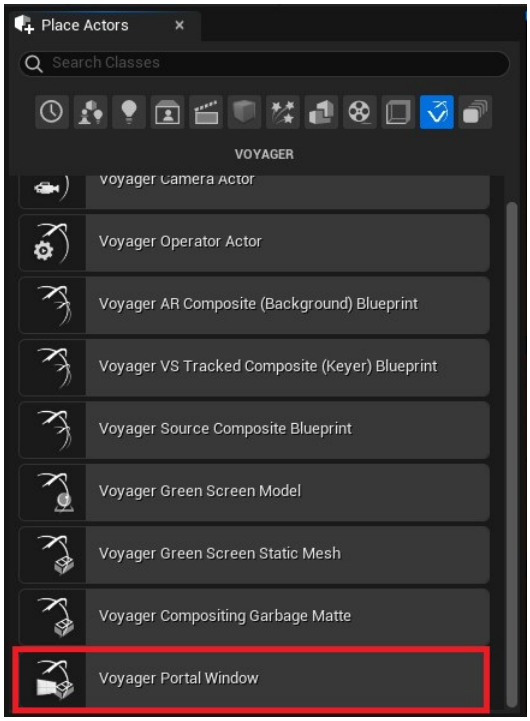
Voyager uses tracked off-axis projection to create a portal effect that gives the impression that you are looking through a window into another space.

The portal effect can also be used to extend the set to give it depth without using a green screen.

Follow the steps outlined in this guide for creating a project from a template or making an existing project compatible with Voyager. Then follow the instructions below to add a portal effect.

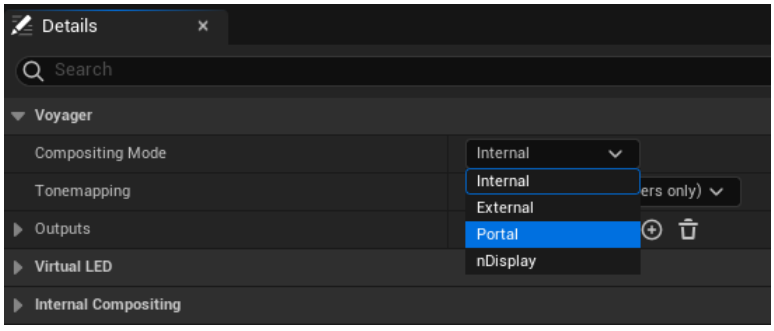
To set up a portal effect:

1. In the **Place Actors** tab, from the **Voyager** category, drag the **VoyagerPortalWindow** actor into the level.



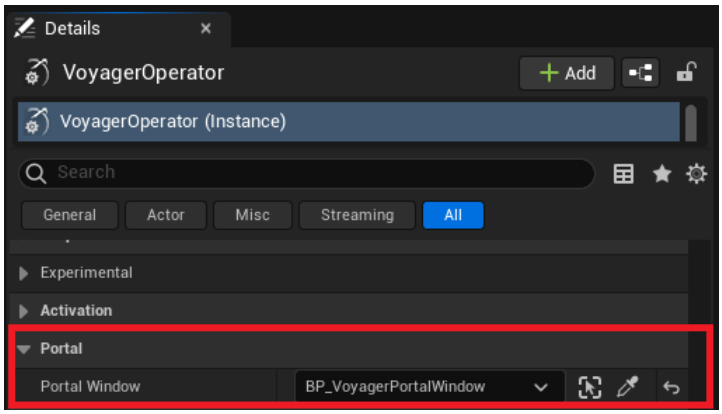
Voyager Portal Window Actor

2. In the **Content Drawer**, navigate to the **VoyagerOperator** actor and double-click on it to open the **Details** editor.
3. Change the **Compositing Mode** to **Portal**, select **Apply** and close the editor.



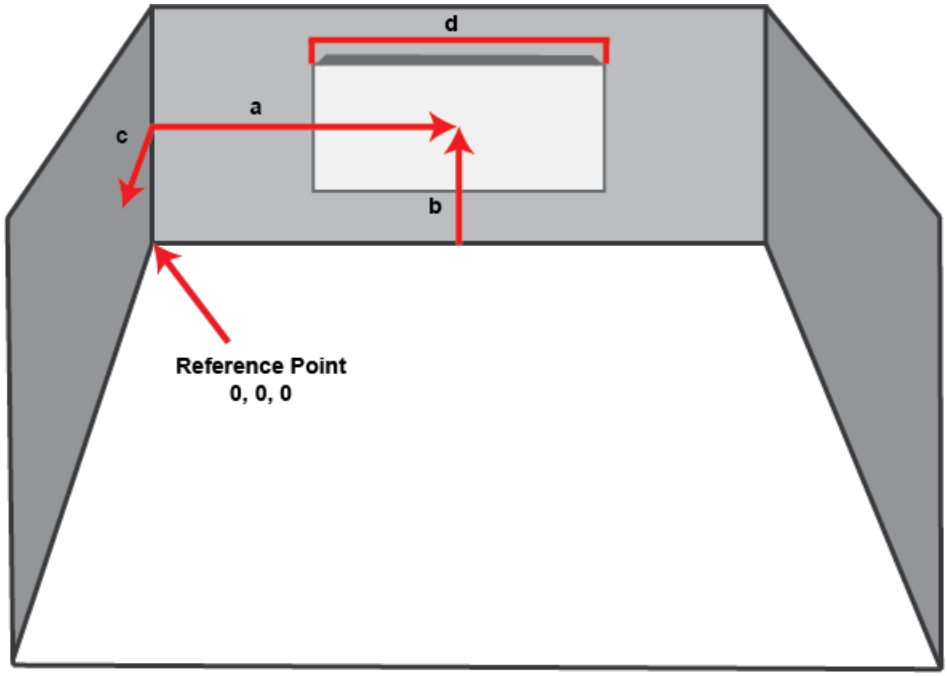
Select Portal Mode

4. In the **Outliner**, select the **VoyagerOperator** and in the **Portal** section of the **Details** tab, from the **Portal Window** drop-down, select the **BP_VoyagerPortalWindow** actor.



Select BP_VoyagerPortalWindow

5. In the **Outliner**, select the **VoyagerCameraActor1** and in the **Transform** section of the **Details** tab, set the Location to **0, 0, 0**.
6. Take the following measurements from the **0, 0, 0** reference point of the room:
   a. Horizontally to the center of the screen.
   b. Vertically from the floor to the center of the screen.
   c. Forwards to the front face of the screen.
   d. The width of the screen.



Measuring to Center of Screen

7. In the **Outliner**, select the **BP_VoyagerPortalWindow** and in the **Transform** section of the **Details** tab:

   a. Enter the measurements for **a**, **b**, and **c** into the appropriate **Location** fields.

   b. Select the lock to the right of the **Scale** field to lock the axis.

   c. Enter the measurement for **d** into the appropriate **Scale** field.

      The height of the screen will automatically be calculated to maintain a 16/9 ratio and be entered into the appropriate field.

   d. Enter a value of **1** into the third **Scale** field.

To position the portal window:

1. In the viewport, select both the **VoyagerCameraActor1** and the **BP_VoyagerPortalWindow** actor.
2. Move both actors together to the position that gives you the desired view.
3. Press **Play** to check the viewpoint.

   Tip:

   In the **VoyagerTracker**, set the **Target Buffer Depth** setting to the lowest value that works for your project.

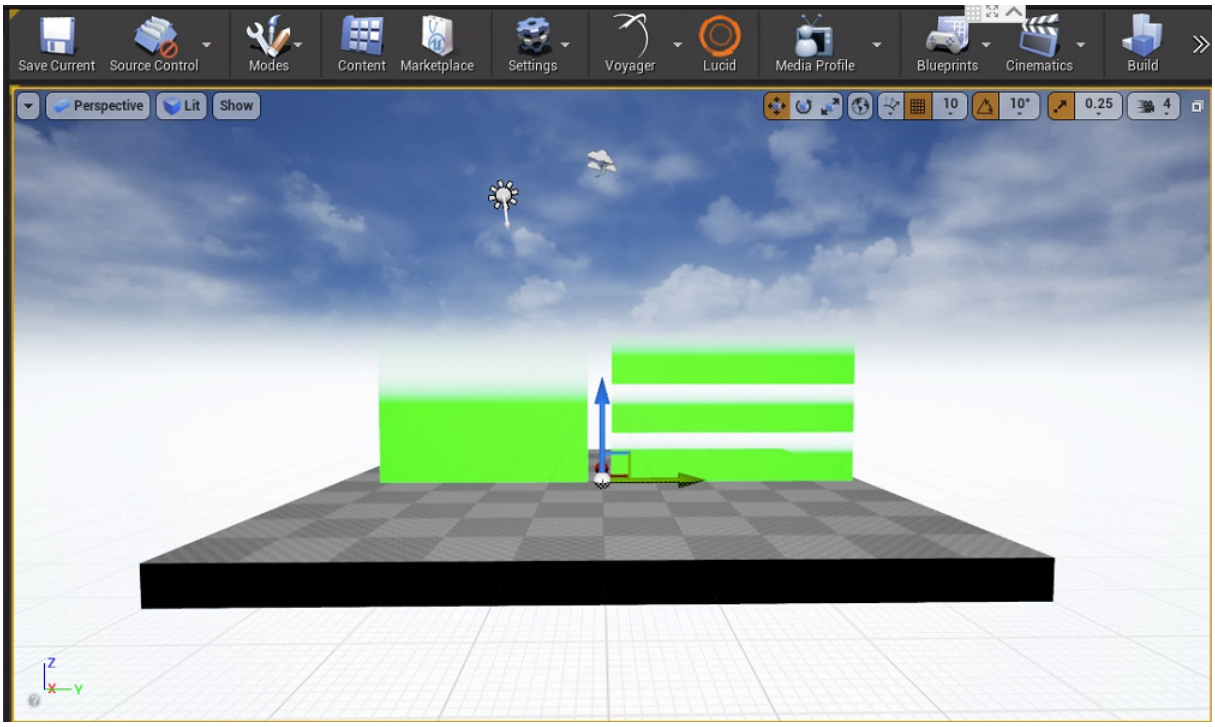
Using the Voyager Green Screen Model

When you're project uses internal compositing, you have the option to add a virtual green screen to define the area on which a video or static image will be displayed.

If you have created your project from a template, the **BP_VoyagerGreenScreen** model is included. If you have not used a template, you'll have to add the model as described below.

You can also use a custom green screen model that you've created in another program. In this case, you'll have to ensure that the mapping of the UV coordinates in the model applies to the static mesh as a whole, rather than to individual sections. This will ensure that any feathering is applied correctly.

In the two examples of the green screen model shown below, the UV coordinates are correctly mapped on the model on the left, but incorrectly mapped on the model on the right. Incorrect mapping results in the feathering being applied to each section of the model instead of to the model as a whole.



UV Mapping Example

To further refine the green screen area, you can add garbage mattes to the scene to hide parts of the physical set that you don't want to see in the output.

You might find it helpful to first open the Voyager Editor Preview Output window. This allows you to see how the set will look in Play mode without actually playing it.

To open the Voyager Editor Preview Output window:

1. In the **Outliner**, select the **VoyagerOperator**.
2. In the **Details** panel, expand the **Editor Preview Output** section and select the **Enable Editor Output Capture**.

   Alternatively, you can select **Start Capture**.
3. To keep the **Preview** window on top, select the **Keep Preview Window on Top** checkbox.

To add the Voyager Green Screen model:

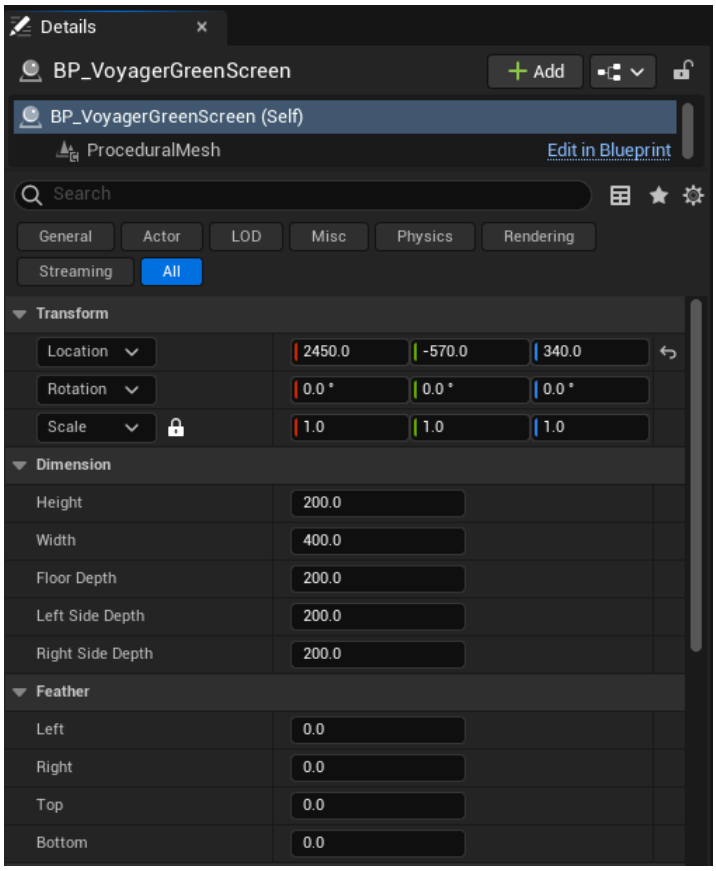
1. In the **Place Actors** tab, select on the **Voyager** group icon.
2. Drag the **Voyager Green Screen Model** into your project scene.

   By default, the model is given the name **BP_VoyagerGreenScreen**.
3. In the **Outliner**, move the **BP_VoyagerGreenScreen** actor into the **Voyager** folder, to be able to find it quickly.
4. In the **Outliner**, select **VoyagerComposite1**.
5. In the **Details** tab, scroll down to the **Internal Compositing** section and select the **Use Physical Set Model** checkbox.

To configure the Voyager Green Screen:

1. In the **Outliner**, select the **BP_VoyagerGreenScreen** actor.
2. Using the **Move** tool, position the **BP_VoyagerGreenScreen** approximately where you want it in the scene.
3. In the **Details** tab, in the **Transform** section, adjust the **Rotation** of the **BP_VoyagerGreenScreen** actor, if necessary.

   Alternatively, you can use the **Rotate** tool in the **Viewport** to adjust the rotation.



Voyager Green Screen Details

4. In the **Dimension** section, increase or decrease the values of each parameter to make the size of the **BP_VoyagerGreenScreen** actor roughly the same size as the physical green screen.

5. In the **Feather** section, increase or decrease the values of each parameter to adjust the blending of each edge of the green screen model.
   - The **Left** feathering is applied from u=0, going towards 1.
   - The **Right** feathering is applied from u=1, going towards 0.
   - The **Top** feathering is applied from v = 0, going towards 1.
   - The **Bottom** feathering is applied from v =1, going towards 0.

To use a custom shape for the Voyager Green Screen model:

1. From the **Place Actors** tab, select and drag a **Voyager Green Screen Static Mesh** actor into the project scene.
2. In the **Outliner**, select the **Voyager Green Screen Static Mesh** actor.
3. In the **Details** tab, scroll down to the **Static Mesh** section and from the drop-down, select your custom green screen model.
4. Then delete the **BP_VoyagerGreenScreen** actor (if using a template).

To add a Voyager Compositing Garbage Matte:

1. In the **Place Actors** tab, select on the **Voyager** group.
2. Drag the **Voyager Compositing Garbage Matte** into your project scene.
3. By default, the matte is given the name **BP_VoyagerCompositingGarbageMatte** and is red.

To configure a Voyager garbage matte:

1. In the **Outliner**, select the **BP_VoyagerCompositingGarbageMatte**.

   If you don't see a red rectangle when the garbage matte is selected, try changing the **Z Rotation** value.
2. With the **BP_VoyagerCompositingGarbageMatte** selected, use the **Move** tool to place the actor on top of the part of the set you want hidden.

   Wherever the garbage matte intersects with the VoyagerComposite1 plane, you won't see that part of the plane.
3. Use the **Scale** tool to adjust the size of the garbage matte, so that it just covers the area to be hidden.
4. In the **Outliner**, move the **BP_VoyagerCompositingGarbageMatte** actor into the **Voyager** folder, to be able to find it quickly.

Voyager Plugins

Voyager contains a number of plugins to make the connections between Voyager and other video production components, including other Voyager plugins. Some are enabled by default, while others need to be enabled.

Enabled by Default:

- Lucid Plugin - Allows Lucid Studio to operate Voyager remotely. See Configuring the Lucid Studio Plugin.
- Matrox Media Player for Voyager - Implements input and ouput using Matrox DSX cards.
- Ross RossTalk™ Plugin - Receives and processes RossTalk™ Commands. See Configuring the RossTalk Plugin.
- Ross Voyager Core Plugin - Provides the shared functionality and common dependencies for other Voyager plugins.
- Ross Voyager Media I/O Framework - Core interfaces and utilities for the Media I/O plugins inside Voyager.
- Ross Voyager Plugin - Handles AR and VS solutions, as well as Virtual LED set extensions.
- Ross Voyager Tracker Plugin - Handles camera tracking.

Disabled by Default:

- PIERO Plugin - Communicates with the PIERO sports graphics analysis tool.
- Ross Adrienne GPIO Plugin - Communicates with a switcher. See Using the Adrienne GPIO Plugin.
- Ross D3 Plugin - Sends commands to D3 TrueControl API.
- Ross DataLinq™ Plugin - The client for the XPression DataLinq server. See Configuring the DataLinq Plugin.
- Ross Tally TSL UMD Plugin - Provides visual cues (in PIE mode) to indicate the status of the Voyager machine, whether it is on air or only in preview mode. It can also send custom TSL UMD messages for general purposes. It supports the TSL 5 protocol which is required for Acuity workflows.
- Ross Voyager API Plugin - Enables Voyager's RESTful API server for third-party integration. See Enabling and Using the Voyager Web API.
- Ross Voyager Charts Plugin - Enables the creation of 3D charts using a DataLinq source. Requires the DataLinq Plugin and a valid DataLinq license. See Using Voyager Charts.
- Ross Voyager Trackless Plugin - Required to use Voyager Trackless for projects with no tracked cameras Voyager Trackless requires an additional license (optional.)
- NDI® IO Plugin - Enables NDI I/O for Voyager. See Enabling the NDI® Media Plugin.

Enabling the Voyager Plugins

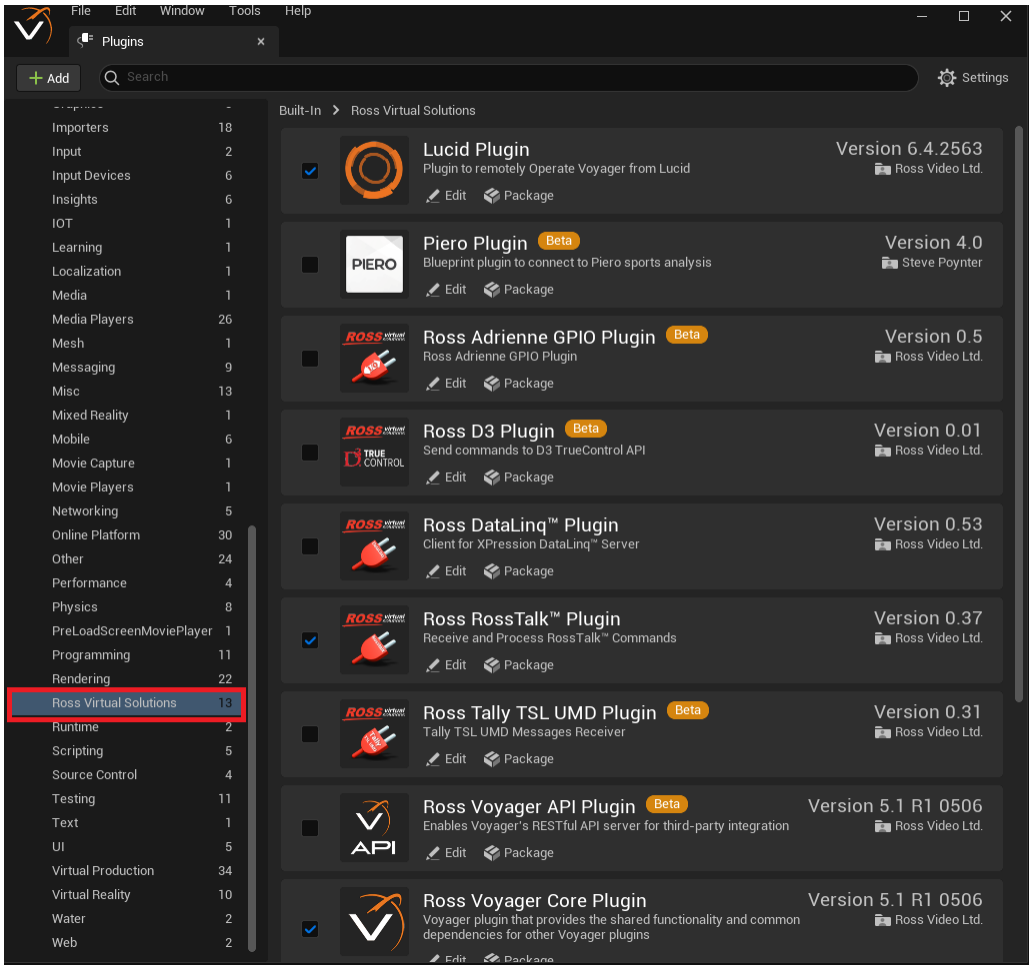
The process is the same to access any of the Ross Virtual Solutions plugins.

To access the Voyager plugins:

1. From the **Edit** menu, select **Plugins**.

   Alternatively, select **Settings > Plugins** from the main toolbar.

2. Scroll down in the list on the left side and select **Ross Virtual Solutions**.



Voyager Plugins

   The list will move to the Voyager plugins.

3. Select the plugin you want to enable or edit.
4. Select the checkbox to the left of the plugin to enable it.
5. Select **Restart Now** when prompted.

Configuring the Lucid Studio Plugin

The Lucid Studio Plugin is the interface between Lucid Studio and Voyager. When running Lucid Studio with the Voyager renderer, you need to set up communication between Lucid Studio and Voyager.

Once communication is established, you'll also be able to use the Lucid Studio logic function to query Voyager and have Voyager return the values that can be set from the **Renderer Logic** function block in Lucid Studio.

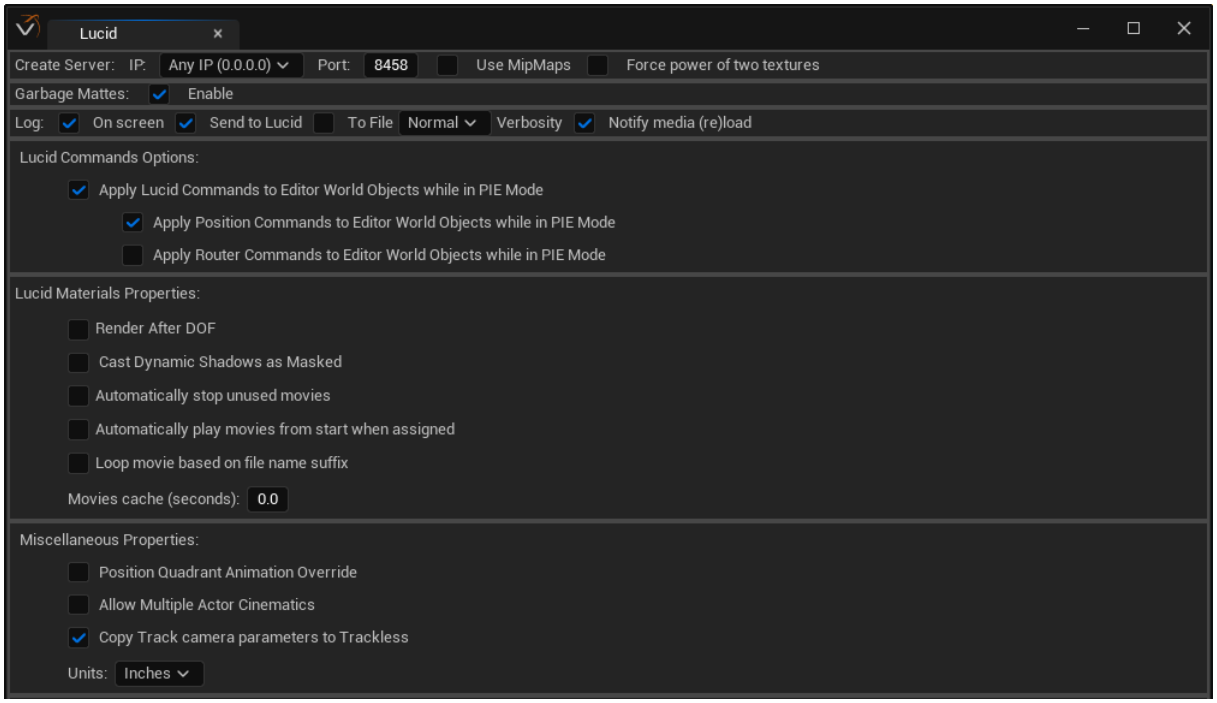
To configure the Lucid Plugin Settings:

1. In Voyager, in the main toolbar, select the **Lucid** icon.

   Alternatively, you can select **Window > Lucid > Lucid Plugin Settings**.

   The Lucid plugin opens.

   Lucid Plugin Settings

2. In the **Create Server** section, from the **IP** drop-down, select the **IP** address that will be used by the Lucid Plugin to listen to Lucid commands.

   You can also select the option **Any IP (0.0.0.0)** which will allow communication from any address.

3. In the **Port** field, enter the port on which the Lucid Studio plugin will listen for Lucid Studio commands.

   This is the same port as is defined when adding a renderer in **Lucid Studio > Server Panel > Server > Add New Element**, in the **Operate Port** field.

   The default port is **8458**.

4. Select the **Use MipMaps** checkbox, to generate mipmaps for image sequences.

   Using mipmaps increases rendering speed and reduces stress on the CPU.

   Default is unchecked.

5. Select the **Force power of two textures** checkbox to convert textures that are not sized to "power of two" dimensions to "power of two" dimensions, so mipmaps can be used.

   This setting is only applicable if **Use MipMaps** is enabled.

6. Select the **Garbage Mattes** checkbox to create garbage mattes in the renderer.

   Garbage mattes are only enabled by this setting. They need to be configured in **Lucid Studio > Track Operate Panel > Garbage Mattes**, to take effect.

   Default is unchecked.

7. In the **Log** section, make the following optional selections:

   - Select the **On screen** checkbox if you want to show the Lucid Studio log on the Unreal Editor screen.
   - Select **Send to Lucid** if you want to send a log to Lucid Studio.

     AND/OR

     Select **To File** to save the log in a file.

     The log will be saved in the project **Saved > Logs** folder.

   - From the **Verbosity** drop-down, select the amount of detail you want to get in the log.
   - Select the **Notify media (re)load** checkbox to notify Lucid Studio when all media have been loaded after reloading sources and every loaded source is logged.

8. In the **Lucid Studio Materials Properties** section, make the following optional selections:

   - Select the **Render After DOF** checkbox to enable **Render After DOF** on Lucid Studio materials.

     ★This requires a restart.

   - Select the **Cast Dynamic Shadows as Masked** checkbox to have Lucid Studio materials cast dynamic shadows as masked.
   - Select the **Automatically stop unused movies** checkbox to stop and reset to the beginning, any movie that is not being used in any actor.
   - Select the **Automatically play movies from start when assigned** checkbox to play movies from the beginning when assigned in Lucid Studio (unless they are already visible in another object).
   - Select the **Loop movie based on file name suffix** checkbox to override Lucid's "**Loop**" command, setting it to true of a movie file ends with "**_LOOP**" or false if a movie file ends with "**_NOLOOP**".
   - In the **Movies cache (seconds)** field, use the arrows to set a time (in seconds) for which movies should be played in the background, on load. This improves the first run in certain (usually high resolution) movies, but it takes longer for the project to fully load.

9. In the **Miscellaneous Properties** section, make the following optional selections:
    - Select the **Position Quadrant Animation Override** checkbox to allow an active animation to be overridden in Lucid Studio with an event that controls the same item.
    - Select the **Allow Multiple Actor Cinematics** to allow running **Sequences** and **Matinees** with actors that are already animated in another cinematic of the same type.
    - Select the **Copy Track camera parameters to Trackless**, if you want any virtual camera in the set to have the same camera properties as the broadcast camera.

      OR

      Clear the checkbox if you want to ensure that the virtual camera(s) you set up in the editor retain their own properties.
10. From the **Units** drop-down, select the unit of length to be used by the Lucid Plugin and close the Lucid plugin.

Configuring the RossTalk Plugin

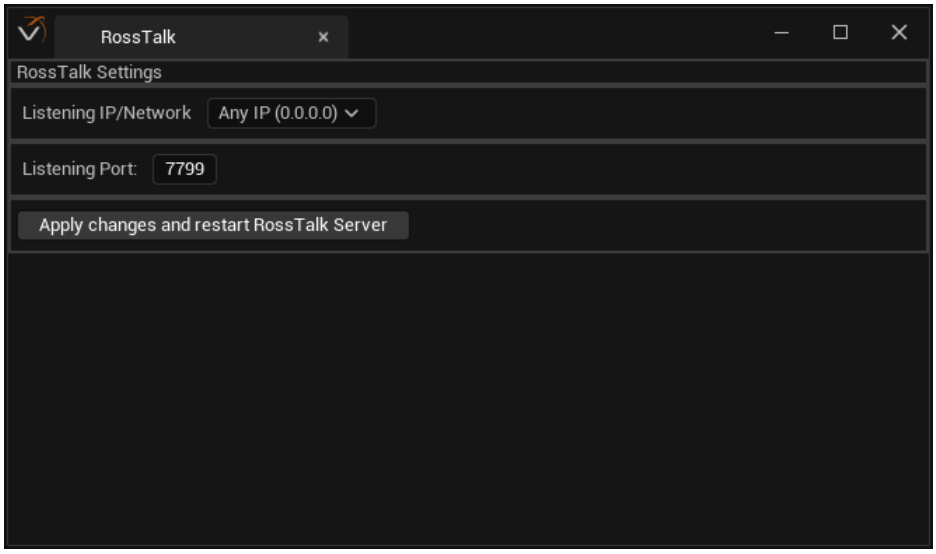
The RossTalk Plugin is enabled by default and can be configured in the Voyager UI. It is not necessary to restart Voyager when making changes to the plugin.

For information on using RossTalk in your projects, see Using RossTalk.

To configure the RossTalk plugin:

1. Select **Window > Voyager > RossTalk Settings**.

   The **RossTalk Settings** tab opens.



   RossTalk Settings

2. From the **Listening IP/Network** drop-down, select the local IP address to be used by the plugin to listen to RossTalk commands.
3. In the **Listening Port** field, enter the number of the **TCP** port to be used by the plugin to listen to RossTalk commands.

   The default port is **7799**.
4. Select **Apply changes and restart RossTalk Server**.
5. Close the **RossTalk Settings** tab.

Configuring the Voyager Web API Plugin

The Voyager Web API is a beta feature that allows an authorized user to access OpenAPI documentation to retrieve detailed information on the processes and to understand and interact with remote engines (for example, for testing purposes) without needing to log into each engine. The machine from which you are accessing the Web API must be on the same network or VPN as the Voyager engines.

The OpenAPI documentation can be accessed at a URL defined by the IP address of the Voyager engine followed by the port number and "openAPI".

(eg. http://xx.xx.xxx.xx:8087/openAPI).

You can also access Swagger, the more user-friendly version of the API using a URL defined by the IP address of the Voyager engine followed by the port number and "swagger".

(eg. http://xx.xx.xxx.xx:8087/swagger).

See Using the Voyager Web API for more information.

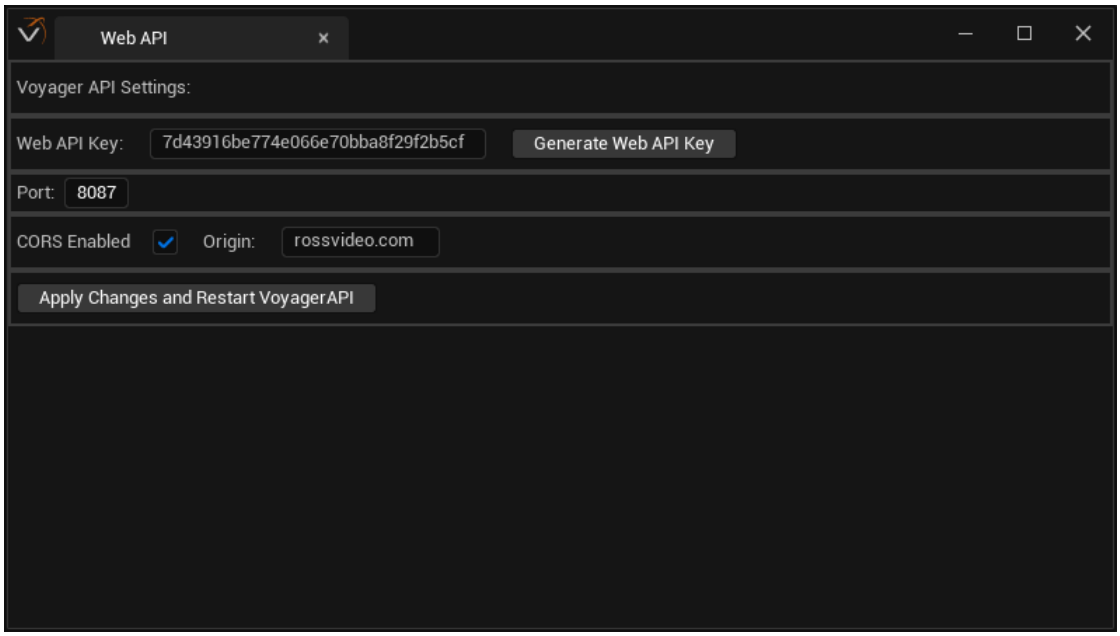
The Voyager Web API is disabled by default. You will need to enable it if you want to use it. See Enabling the Voyager Plugins for instructions. You will need to restart Voyager after enabling the plugin.

To configure the Voyager Web API:

1. Select **Window > Voyager > Web API Settings**.

   In the **Web API** editor the **Web API Key** is automatically entered when the API plugin is enabled.

   The **Web API Key** is a unique identifier that is randomly generated on every new Voyager machine and can be changed manually if necessary.



   Web API Settings

2. Then enter the **Port** number on which the API will communicate with the Voyager engines.

   The default **Port** number is **8087**, but if this port is being used by something else in your network, you can change it to a free port.

3. Make sure that the **Port** is enabled in the firewall.

   See Appendix A: Enabling a Port Number in the Firewall for instructions.

4. Select the **CORS Enabled** checkbox, to restrict access to the **WebAPI** to machines running on the domain specified in the **Origin** field (replace the asterisk in the field with the domain name).

   Disabled by default.

   ★If you select the **CORS Enabled** checkbox and just leave the asterisk in the **Origin** field, you are allowing anyone to access the Web API.

5. Then select **Apply Changes and Restart VoyagerAPI**.

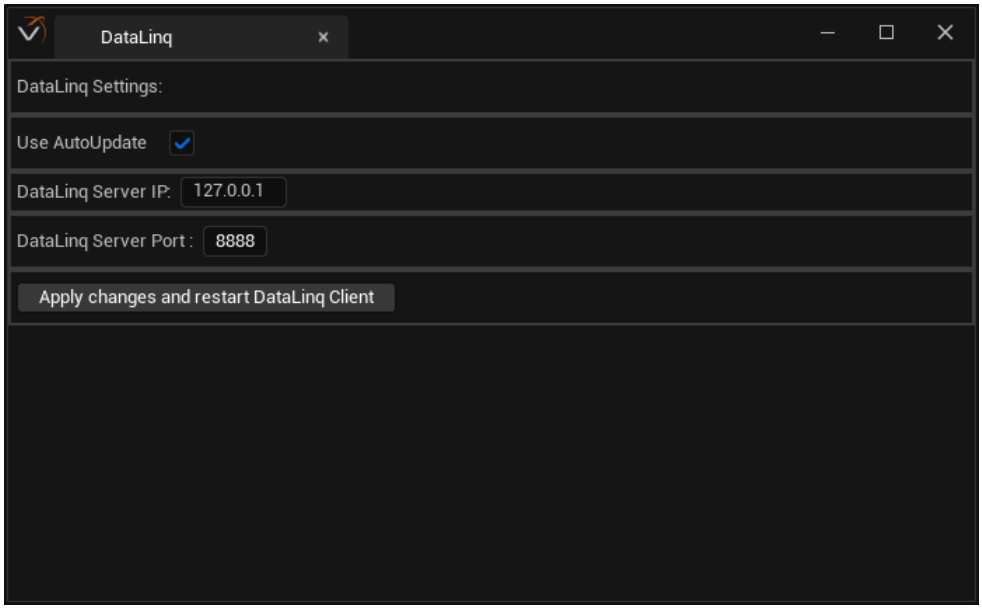
Configuring the DataLinq Plugin

The DataLinq plugin is disabled by default and allows you to connect to a DataLinq server and make DataLinq data available to Voyager. See Enabling the Voyager Plugins for instructions on enabling the DataLinq Plugin. You will need to restart Voyager after enabling the plugin.

To configure the DataLinq plugin:

1. In the **Main** menu, select **Window > Voyager > DataLinq Settings**.

   The **DataLinq Settings** tab opens.



   DataLinq Client Settings

2. Select the **AutoUpdate** checkbox to allow automatic update of the plugin.
3. In the **DataLinq Server IP** field, enter the **IP** address to be used by the plugin to connect to the DataLinq server.
4. In the **DataLinq Server Port** field, enter the port to be used by the plugin to connect to the DataLinq server.

   The default port is **8888**.

5. Select **Apply changes and restart DataLinq Client**.
6. Close the **DataLinq Settings** tab.

Enabling the NDI® Media I/O Plugin

The NDI® Media I/O plugin allows you to create NDI inputs and outputs for Voyager. The plugin is disabled by default.

To enable the NDI I/O Plugin:

1. Select **Edit > Plugins**.
2. In the **Search** field, type `NDI`.
3. Select the **NDI IO Plugin**.
4. When prompted, restart Voyager.

For information about using NDI in your Voyager project, see Using NDI Media I/O.

Using the Adrienne GPIO Plugin

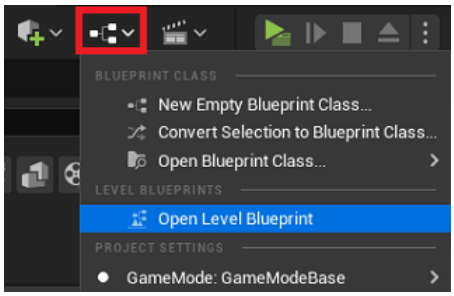
Voyager contains blueprint nodes that read and write the GPI state to the Adrienne card. Selecting a virtual camera relays a message to a switcher (such as Ross Carbonite) which triggers the physical camera switch.

You will need to update the Adrienne Driver 1.0.1.1 to version 5/29/12 and then enable the Adrienne Plugin. For instructions on enabling the Adrienne Plugin, see Enabling the Voyager Plugins.

You can use the Adrienne GPIO Plugin to trigger Voyager project events from a physical signal or to trigger physical events from Voyager.

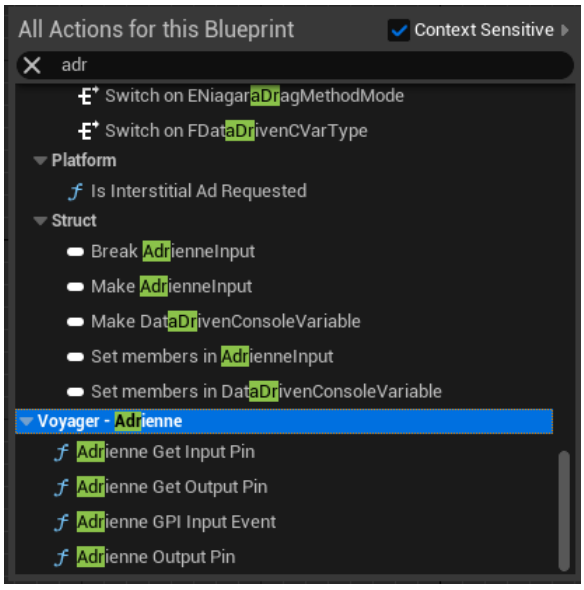
To use the Adrienne GPI nodes:

1. In Voyager, in the main toolbar, select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.



Open Level Blueprint

2. Right-click in the blueprint, and in the **All Actions for this Blueprint** search field, begin typing `Adrienne`.



Adrienne GPI Nodes

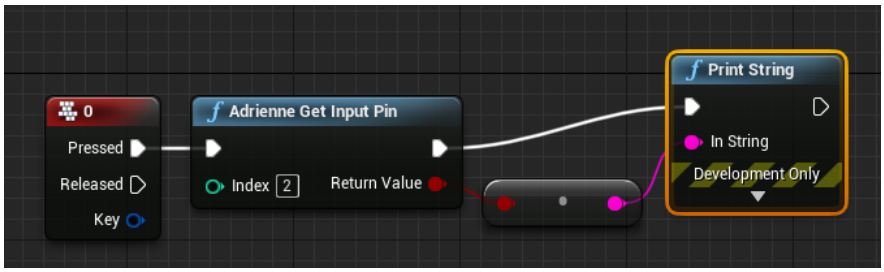
3. Select the node you need.

   The options are:

   Adrienne Get Input Pin - On a key press, retrieves the current state of the selected input pin.

   Enter the pin number in the **Index** field and connect the node to a **Print String** node to generate a **True** or **False** indicator.

   Example:

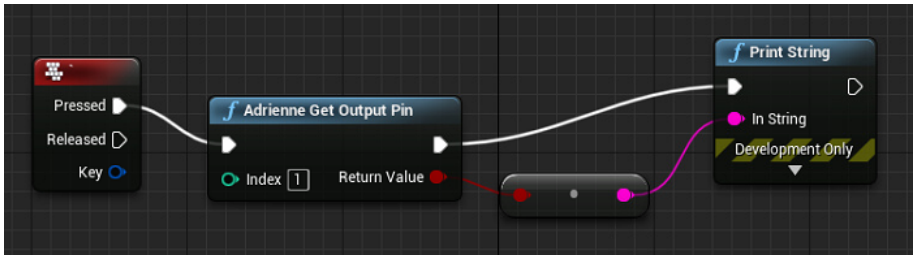


Adrienne Get Input Pin Node

   Adrienne Get Output Pin - On a key press, retrieves the current state of the selected output pin.

   Enter the pin number in the **Index** field and connect the node to a **Print String** node to generate a **True** or **False** indicator.

   Example:

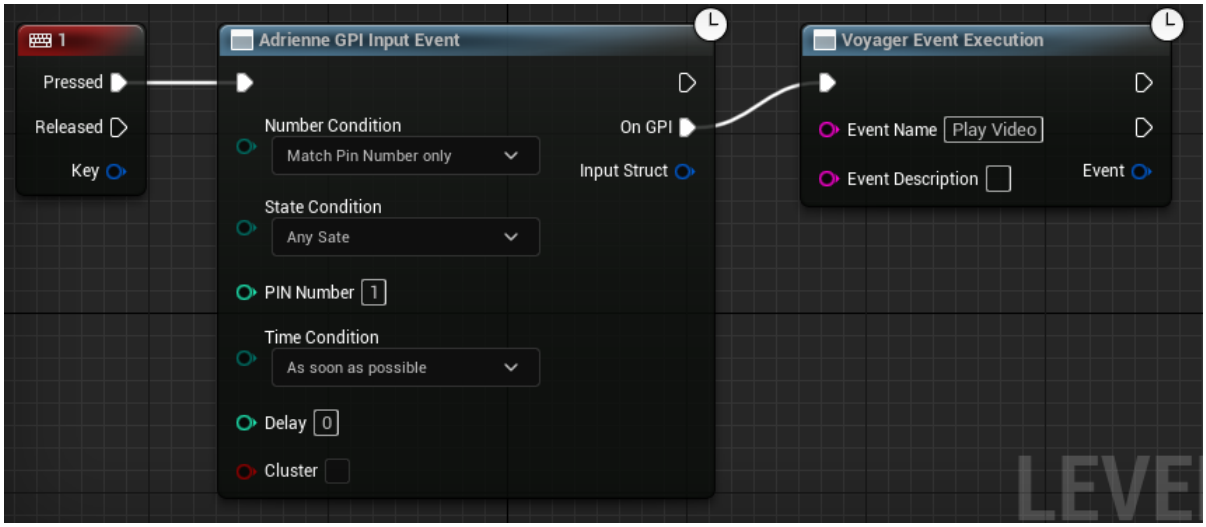


Adrienne Get Output Pin Node

   Adrienne GPI Input Event - Triggers an event when an input state changes and the conditions are met.

Condition	Description	Additional Parameters
Number Condition	Any Pin Match Pin Number only	None Enter pin number in **Pin** field
State Condition	Any State True Only False Only	None " "
Time Condition	As soon as possible After Delay in milliseconds After Delay in frames	None Add milliseconds in **Delay** field Add frames in **Delay** field
Cluster	When selected, the triggered event will occur in all Voyager engines in an nDisplay setup.	None

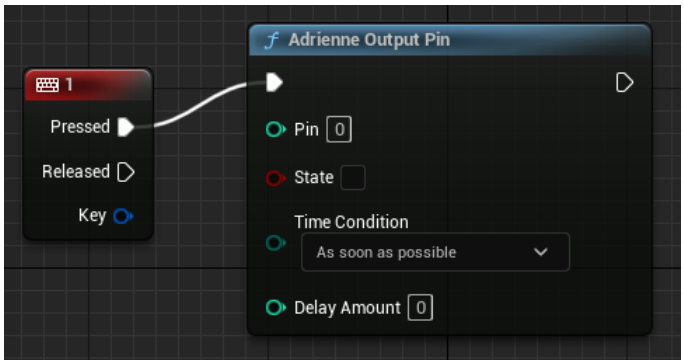
Example:



Adrienne GPI Input Event Node

Adrienne Output Pin - When triggered by an event, triggers a physical reaction (e.g., a set component moves).

Example:



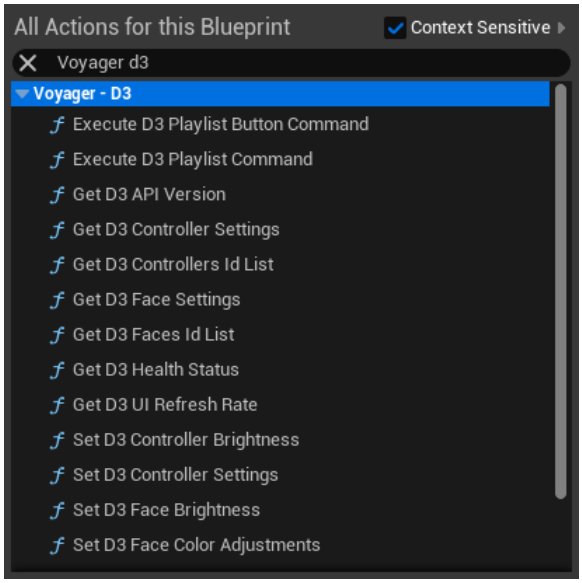
Adrienne Output Pin

Using the D3 Plugin

The D3 plugin allows you to send commands to the D3 TrueControl API. For instructions on enabling the D3 Plugin, see Enabling the Voyager Plugins.

To access the Voyager D3 nodes:

1. Select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.
2. Right-click in the blueprint and in the **Search** field, begin typing `Voyager D3`.



Voyager D3 Blueprints

3. Select the node you want to use.

   For information on creating blueprints for each node, see the following sections:

   Controller Settings — controls the settings of the LED panel

   Face Settings — controls individual screens (faces) that make up a large LED display

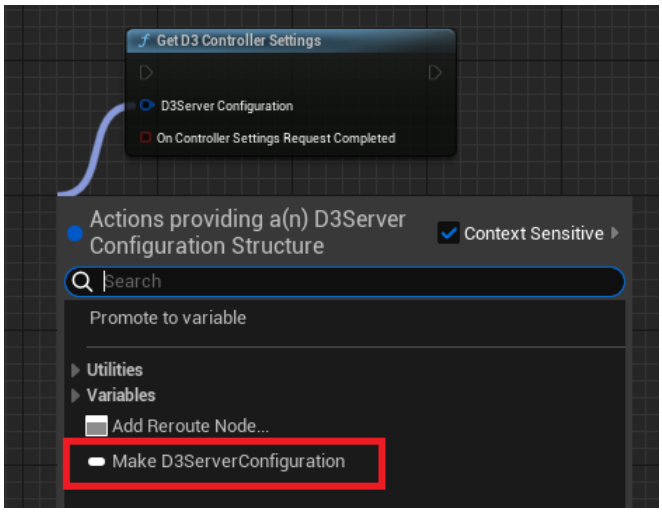
   Playlist Commands — controls the playout of a video playlist

Controller Settings

Use the **Controller** settings to make changes in the device that controls the entire LED panel.

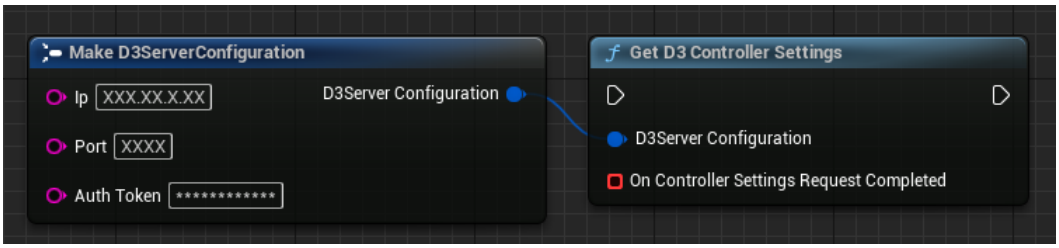
To begin creating a blueprint for any D3 Controller Settings action:

1. Select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.
2. Right-click in the blueprint and in the **Search** field, begin typing `D3` and from the results, select the **D3** node you want.
3. In any **D3** node, left-click and drag off from the **D3Server Configuration** pin and from the **Actions providing a(n) D3Server Configuration Structure** menu, select **Make 3DServerConfiguration**.



Make D3ServerConfiguration

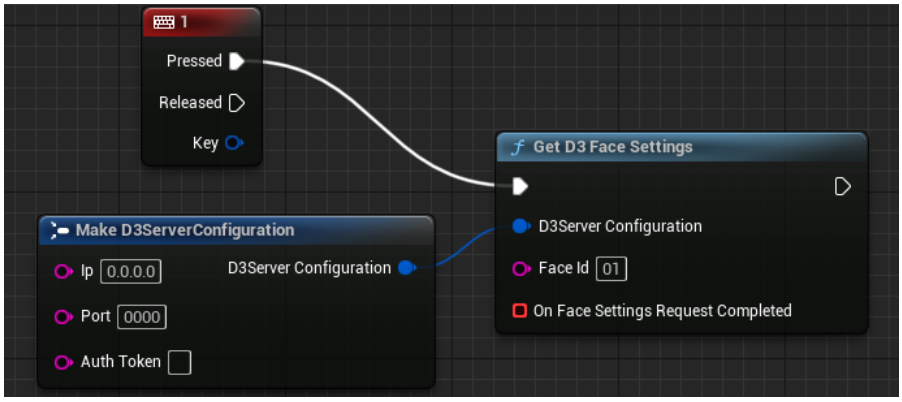
4. In the **Make D3ServerConfiguration** node, do the following:
   - In the **Ip** field, enter the IP address of the D3 server.
   - In the **Port** field, enter the port number for the D3 server.
   - In the **Auth Token** field, enter your token number.



D3 Server Configuration

5. Right-click in the blueprint and select a trigger node (e.g., a button press node, or a RossTalk GPI Event node).

6. Connect the **Output** pin of the trigger node to the **Input** pin of the **D3** node.



Add Trigger

7. Add **Feedback** nodes as follows:

   a. In the D3 node, left-click and drag off from the **Request Completed** pin, begin typing `add custom event` and select the **Add Custom Event** node.

   The **Custom Event** node will differ depending on the D3 node to which it is connected.

   b. Give the **Custom Event** node a unique name that corresponds to the D3 node.

   c. In the **Custom Event** node, left-click and drag off from the **Success** pin, begin typing `branch` and select the **Branch** node.

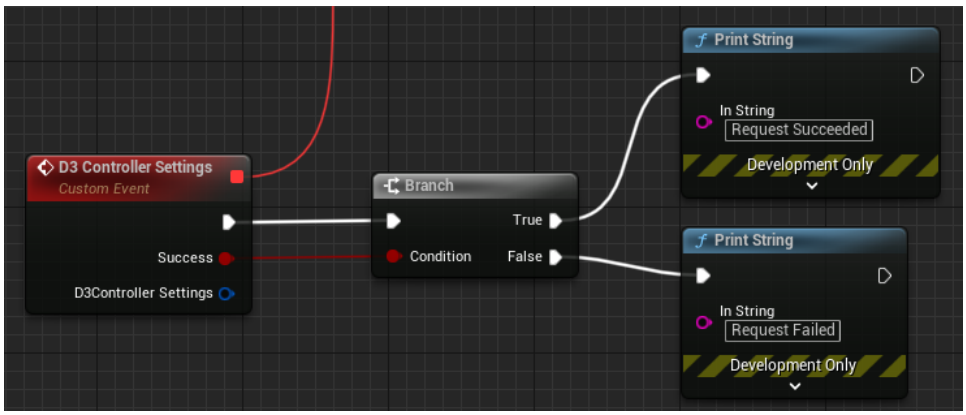
   The **Output** pin of the **Custom Event** node is automatically connected to the **Input** pin of the **Branch** node.

   d. In the **Branch** node, left-click and drag off from the **True** pin, begin typing `print string` and select the **Print String** node.

   e. In the **Print String** node, in the **In String** field, enter some text to indicate success (e.g., Request Succeeded).

   f. In the **Branch** node, left-click and drag off from the **False** pin, begin typing `print string` and select the **Print String** node.

   g. In the **Print String** node, in the **In String** field, enter some text to indicate failure (e.g., Request Failed).



D3 Feedback Nodes

When the command is executed, the success or failure of the command is indicated.

8. Continue with the procedure for the specific D3 blueprint you want.

   Get D3 API Version

   Get D3 Controller Settings

   Get D3 Controllers Id List

   Set D3 Controller Brightness

   Get D3 Health Status

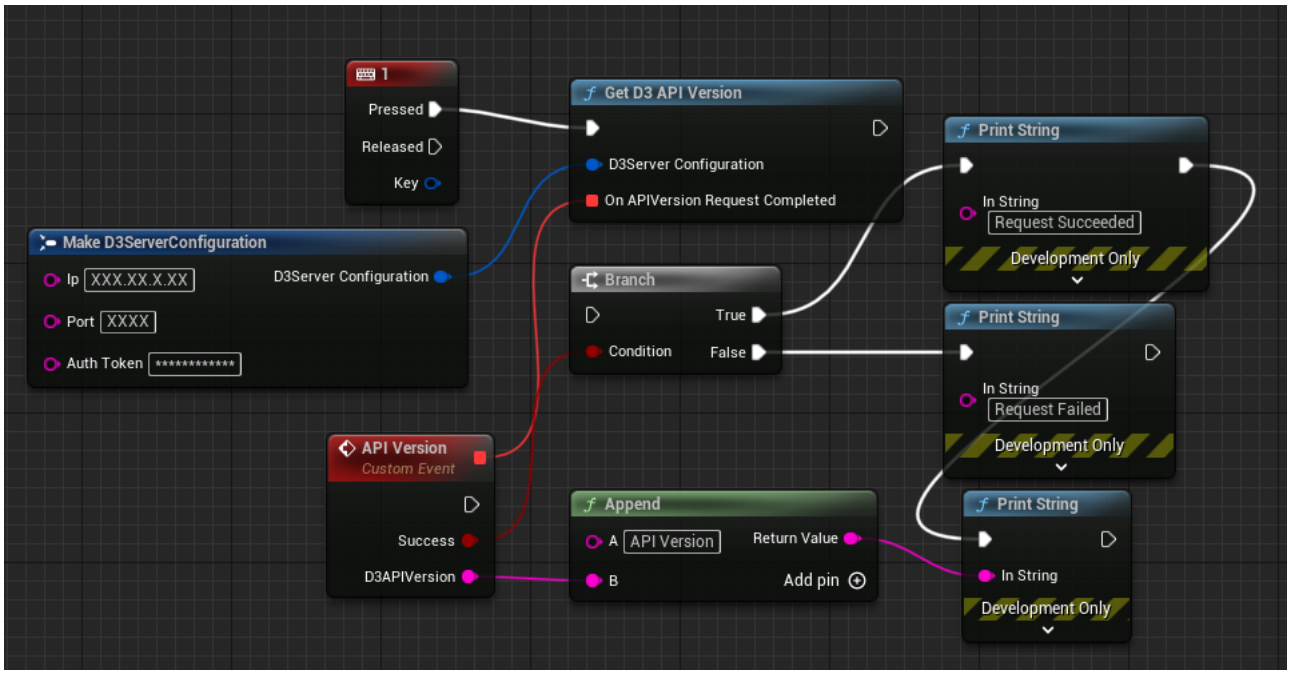
   Set D3 Selected Scene

   Get D3 UI Refresh Rate

To create a Get D3 API Version blueprint:

1. In an empty area of the blueprint, right-click and begin typing `string append` and select the **Append** node.
2. Connect the **D3APIVersion** pin of the **Custom Event** node to the **B Input** pin of the **Append** node.
3. In the **Append** node, in the **A Input** field, enter name of the custom event (e.g., API Version).
4. In the **Append** node, left-click and drag off the **Return Value** pin, begin typing `print string` and select the **Print String** node.
5. Connect the **Output** pin of the **Print String** node connected to the **True** pin of the **Branch** node to the **Input** pin of the new **Print String** node.

   Your blueprint will be similar to the blueprint below:



Get D3 API Version Blueprint

To create a Get D3 Controller Settings blueprint:

1. In the **Custom Event** node, left-click and drag off from the **D3Controller Settings** pin and select the **Break D3 Controller Settings** node.
2. Select the arrow at the bottom of the **Break D3ControllerSettings** node to expand it.
3. In the **Break D3Controller Settings** node, decide which settings you want to get and do the following:
   - For each setting that is a string, right-click in an empty part of the blueprint, begin typing `string append` and select the **Append** node.

     Connect the setting pin to the **B Input** pin In the **Append** node.

     In the **Append** node, in the **A Input** field, enter the name of the setting to which it is connected.

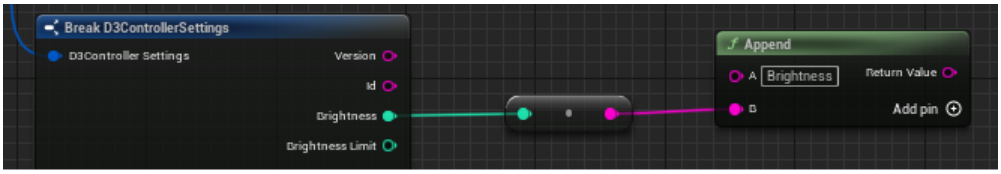
     Add Append Node - String

   - For each setting that is an integer, left-click and drag off from the pin, begin typing `string append` and select the **Append** node.

     The **To String (Integer)** node is automatically inserted between the setting pin and the **Append** node.

     Disconnect the **Output** pin of the **To String (Integer)** node from the **A Input** pin of the **Append** node and connect it instead to the **B Input** pin.

     In the **Append** node, in the **A Input** field, enter the name of the setting to which it is connected.

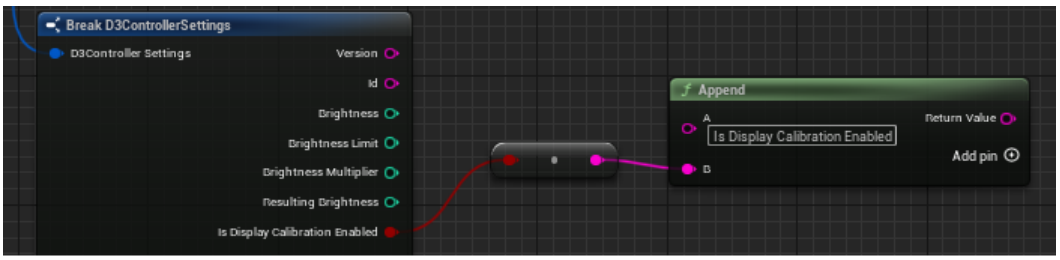
     Add To String (Integer) Node and Append Node

   - For each setting that is a Boolean, left-click and drag off from the pin, begin typing `string append` and select the **Append** node.

     The **To String (Boolean)** node is automatically inserted between the setting pin and the **Append** node.

     Disconnect the **Output** pin of the **To String (Boolean)** node from the **A Input** pin of the **Append** node and connect it instead to the **B Input** pin.

     In the **Append** node, in the **A Input** field, enter the name of the setting to which it is connected.

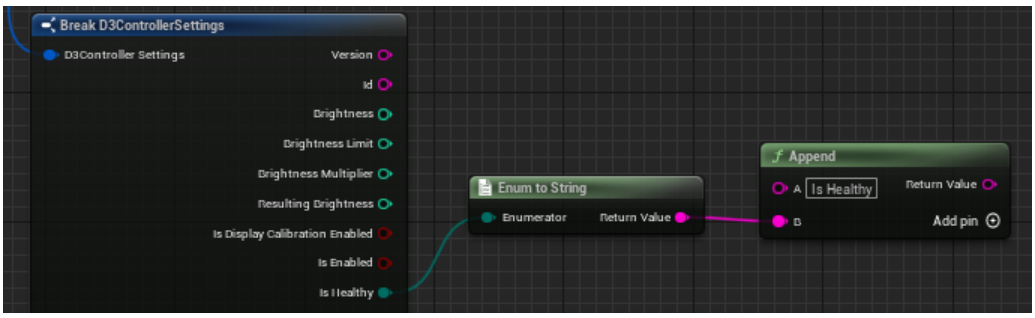
     Add To String (Boolean) Node and Append Node

- For each setting that is an enum, left-click and drag off from the pin, begin typing `string append` and select the **Append** node.

  The **Enum to String** node is automatically inserted between the setting pin and the **Append** node.

  Disconnect the **Return Value** pin of the **Enum to String** node from the **A Input** pin of the **Append** node and connect it instead to the **B Input** pin.

  In the **Append** node, in the **A Input** field, enter the name of the setting to which it is connected.



Add To String (Enum) Node and Append Node

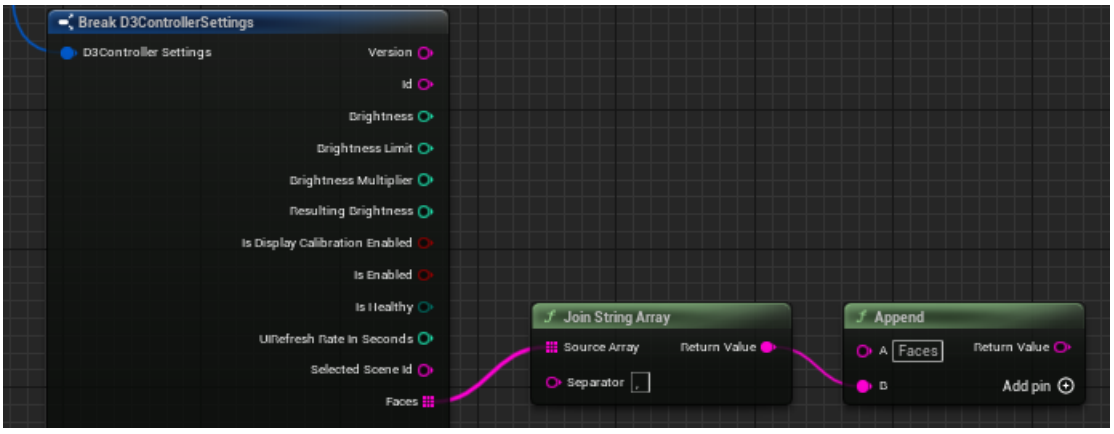
- For each setting that is an array, begin typing join string array and select the **Join String Array** node.

  Right-click in an empty part of the blueprint, begin typing `string append` and select the **Append** node.

  Connect the **Return Value** pin of the **Join String Array** node to the **B Input** pin of the **Append** node.

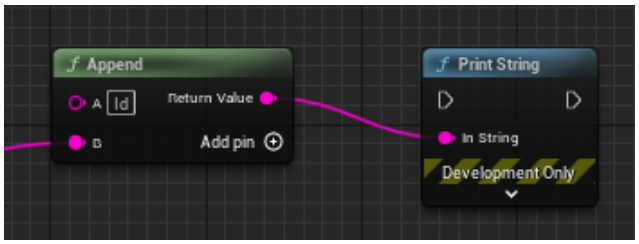
  In the **Join String Array** node, in the **Separator** field, enter a character (e.g., comma) to separate the items in the resulting list.

  In the **Append** node, in the **A Input** field, enter the name of the setting to which it is connected.



Add Join String Array Node and Append Node

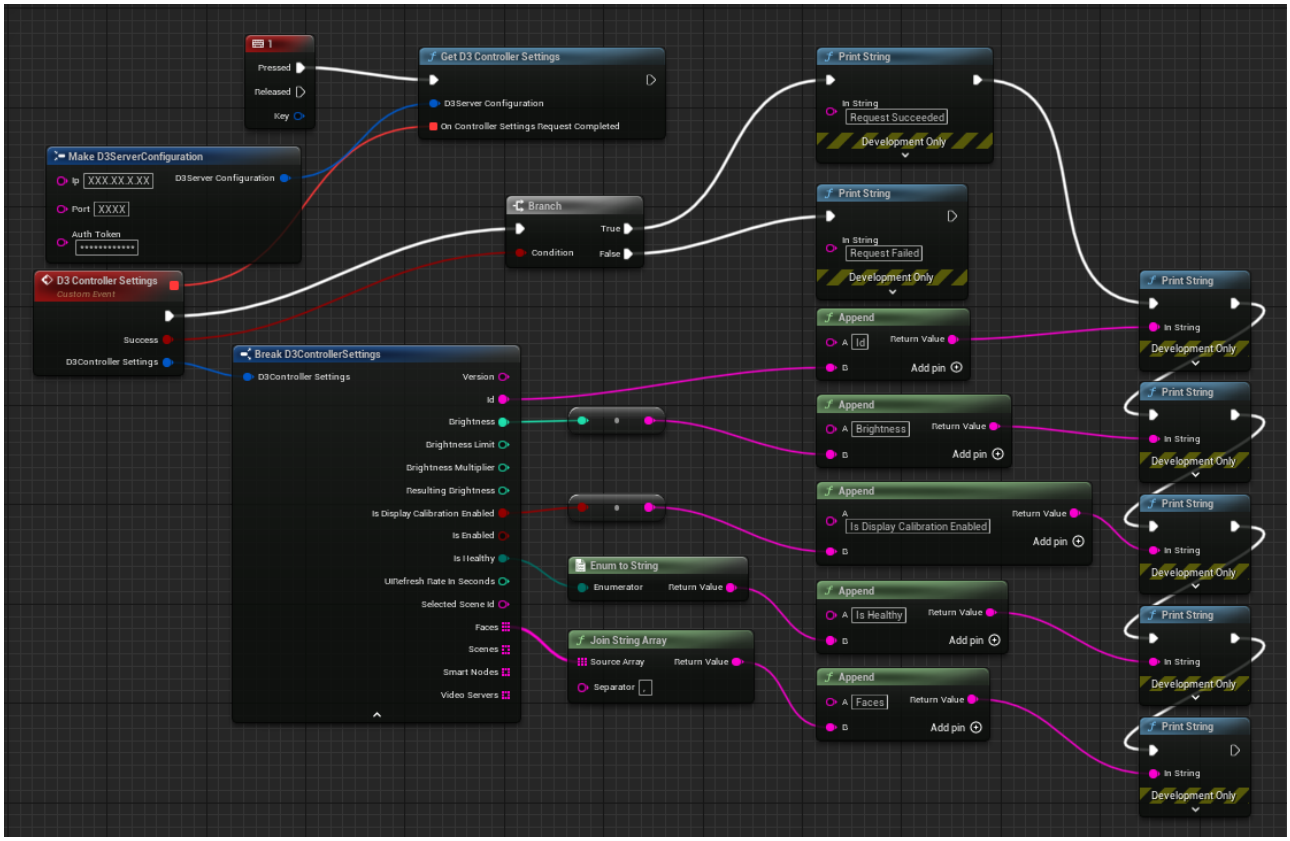
4. Now, left-click and drag off the **Return Value** pin of each **Append** node, begin typing `print string` and select the **Print String** node.



Add Print String Node

5. Connect the **Output** pin of the **Print String** node that is connected to the **True** pin of the **Branch** node, to the **Input** pin of the first setting **Print String** node.
6. Connect the **Output** pin of the first setting **Print String** node to the **Input** pin of the second setting **Print String** node.
7. Continue connecting the **Print String** nodes until they are all connected.

   Your blueprint will be similar to the blueprint below:

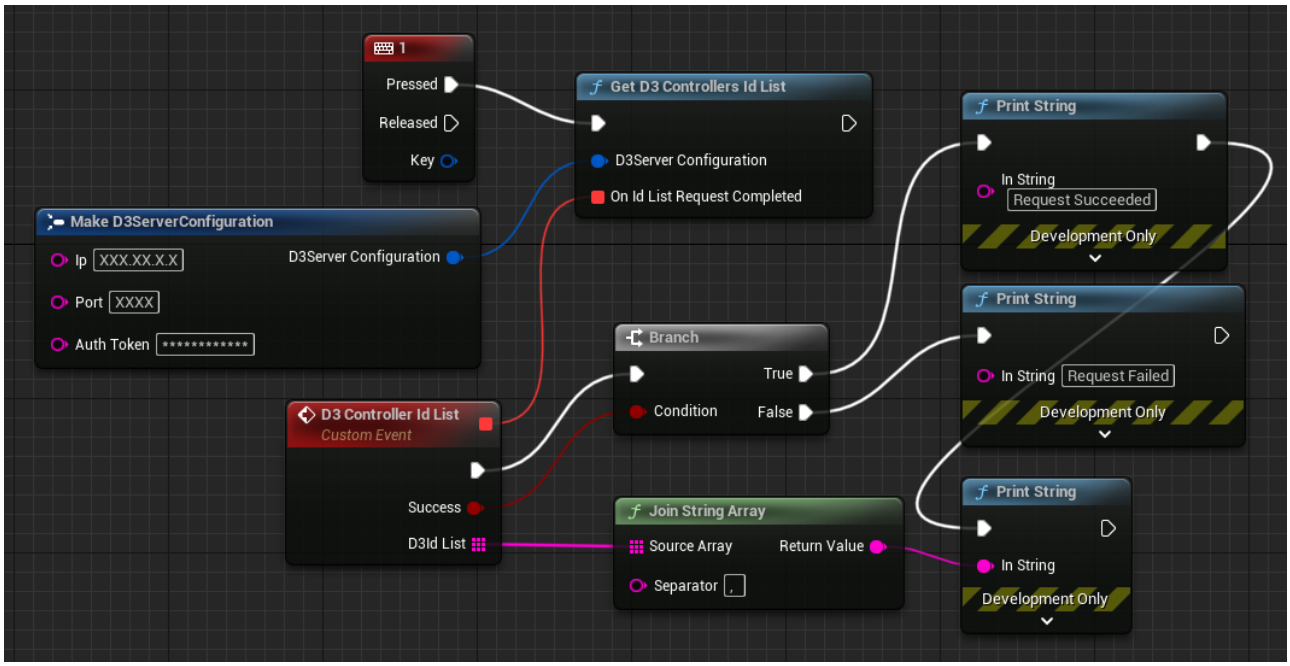


Get D3 Controller Settings Blueprint

To create a Get D3 Controllers Id List blueprint:

1. In the **Custom Event** node, left-click and drag off the **D3Id List** pin, begin typing `join string array` and select the **Join String Array** node.
2. In the **Join String Array** node, in the **Separator** field, enter a character (e.g., comma) to separate the items in the resulting list.
3. Then left-click and drag off the **Return Value** pin, begin typing `print string` and select the **Print String** node.
4. Connect the **Output** pin of the **Print String** node that is connected to the **True** pin of the **Branch** node, to the **Input** pin of the new **Print String** node.

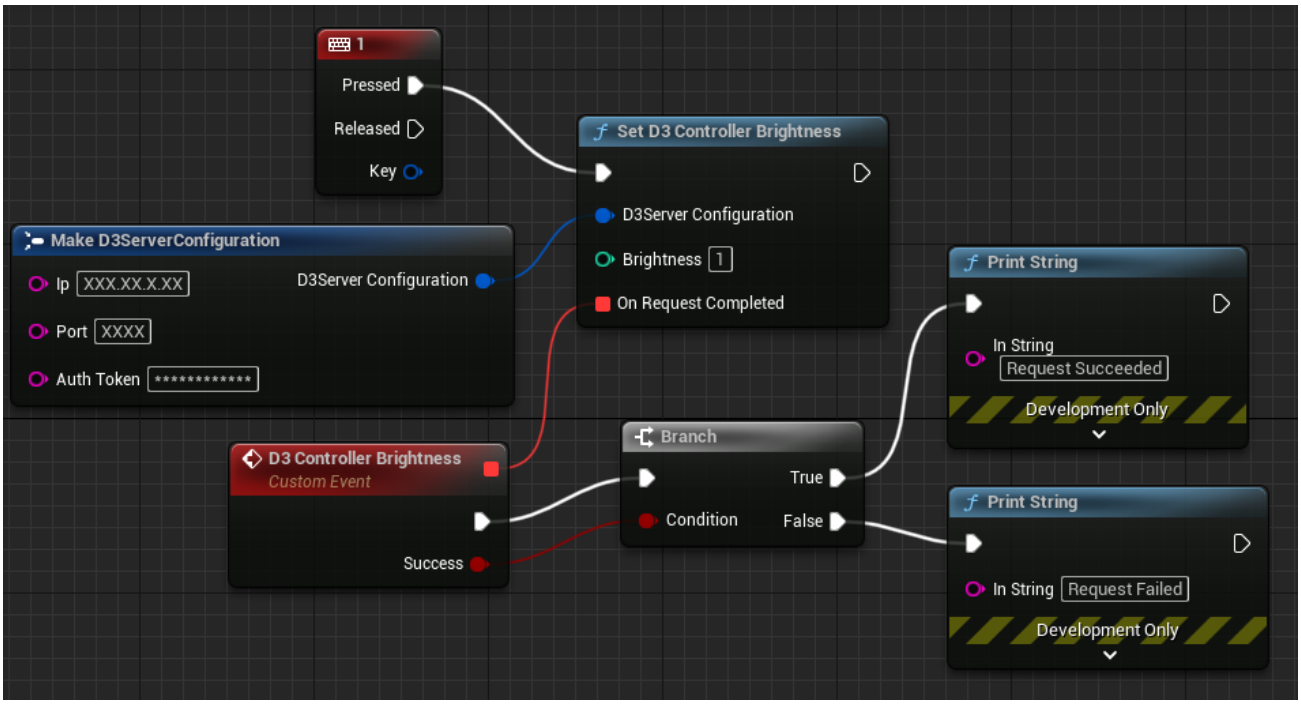
   Your blueprint will be similar to the blueprint below:



Get D3 Controllers Id List Blueprint

To create a Set D3 Controller Brightness blueprint:

- In the **Set D3 Controller Brightness** node, in the **Brightness** field, enter a value between **0 - 100**.



Set D3 Controller Brightness Blueprint

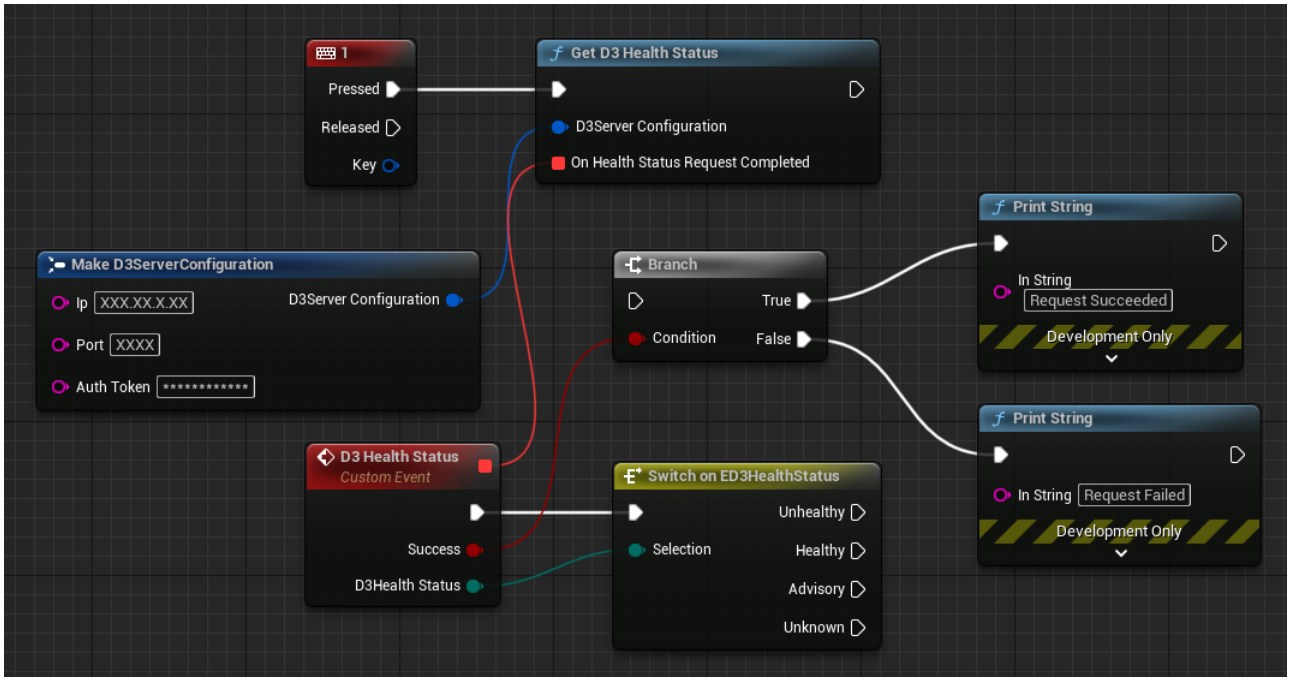
To create a Get D3 Health Status blueprint:

1. In the **Custom Event** node, left-click and drag off the **D3Health Status** pin, begin typing switch and select the **Switch on ED3HealthStatus** node.

   The **Output** pin of the **Custom Event** node is automatically connected to the **Input** pin of the **Switch on ED3HealthStatus** node.

2. Then connect each of the **Status** pins to nodes that will initiate the desired response to the status.

   Your blueprint will be similar to the blueprint below:

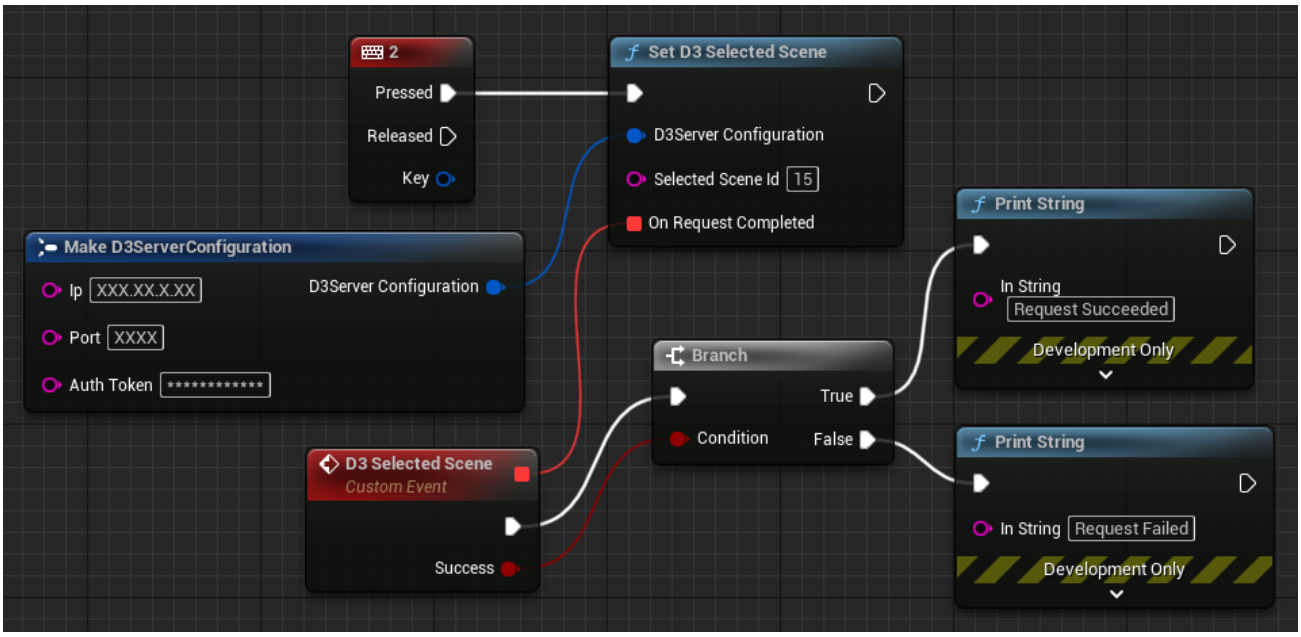


Get D3 Health Status Blueprint

To create a Set D3 Selected Scene blueprint:

- In the **Set D3 Selected Scene** node, in the **Selected Scene Id** field, enter the number of the selected scene.

  Your blueprint will be similar to the blueprint below:



Set D3 Selected Scene Blueprint

To create a Get D3 UI Refresh Rate blueprint:

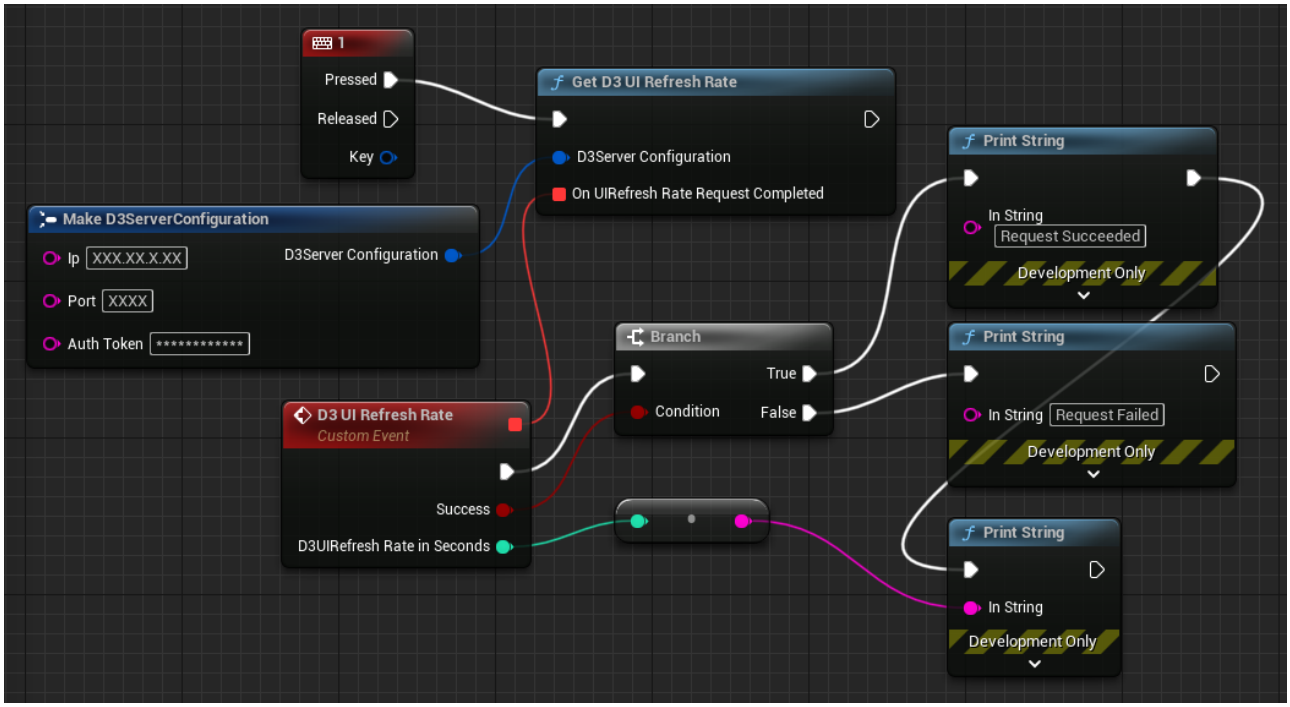
1. In the **Custom Event** node, left-click and drag off the **D3UIRefresh Rate in Seconds** pin, begin typing `print string` and select the **Print String** node.

   If you don't get any results, deselect the **Context Sensitive** checkbox and retry.

   The connecting node, **To String (Integer)** will be inserted automatically between the **Custom Event** node and the **Print String** node.

2. Connect the **Output** pin of the **Print String** node that is connected to the **True** pin of the **Branch** node, to the **Input** pin of the new **Print String** node.

   Your blueprint will be similar to the blueprint below:



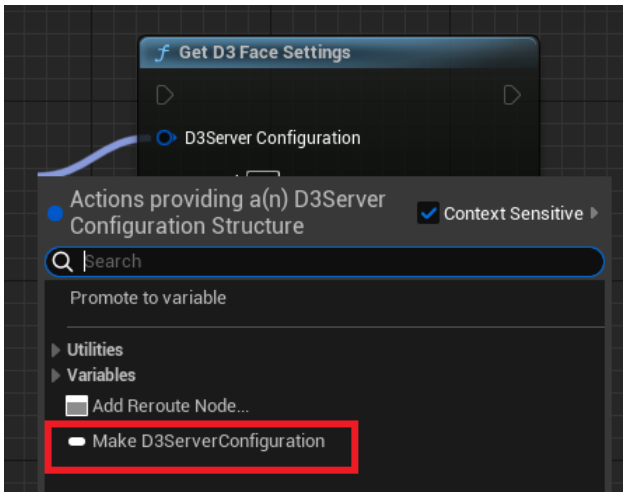
Get D3 UI Refresh Rate Blueprint

Face Settings

Use the **D3 Face Settings** nodes to get and set data for an individual display screen in an LED panel. The first procedure described below applies to all of the nodes. The procedures following the first one pertain to specific nodes.

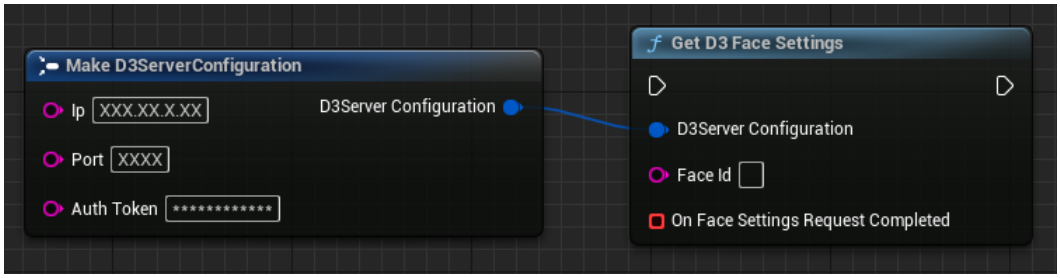
To begin creating a blueprint for any D3 Face Settings action:

1. Select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.
2. Right-click in the blueprint and in the **Search** field, begin typing `D3` and from the results, select the D3 node you want.
3. In any **D3** node, left-click and drag off from the **D3Server Configuration** pin and from the **Actions providing a(n) D3Server Configuration Structure** menu, select **Make 3DServerConfiguration**.



Make D3ServerConfiguration

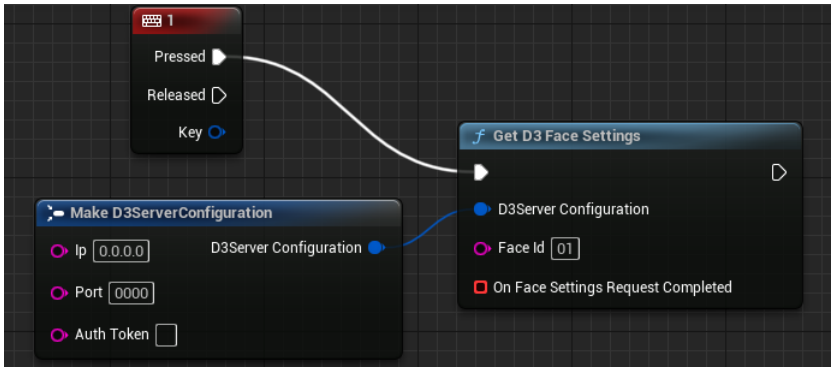
4. In the **Make D3ServerConfiguration** node, do the following:
   - In the **Ip** field, enter the IP address of the D3 server.
   - In the **Port** field, enter the port number for the D3 server.
   - In the **Auth Token** field, enter your token number.



D3 Server Configuration

5. Right-click in the blueprint and select a trigger (e.g., a button press, or a RossTalk GPI Event node).

6. Connect the **Output** pin of the trigger node to the **Input** pin of the **D3** node.



Add Trigger

7. In the **D3** node, in the **Face Id** field, enter the identification number of the display screen for which to get or set data.

   There is no **Face Id** field in the **Get D3 Faces Id List** node.

8. Add **Feedback** nodes as follows:

   a. In the **D3** node, left-click and drag off from the **Request Completed** pin, begin typing `add custom event` and select the **Add Custom Event** node.

      The **Custom Event** node will differ depending on the **D3** node to which it is connected.

   b. Give the **Custom Event** node a unique name that corresponds to the **D3** node.

   c. In the **Custom Event** node, left-click and drag off from the **Success** pin, begin typing `branch` and select the **Branch** node.

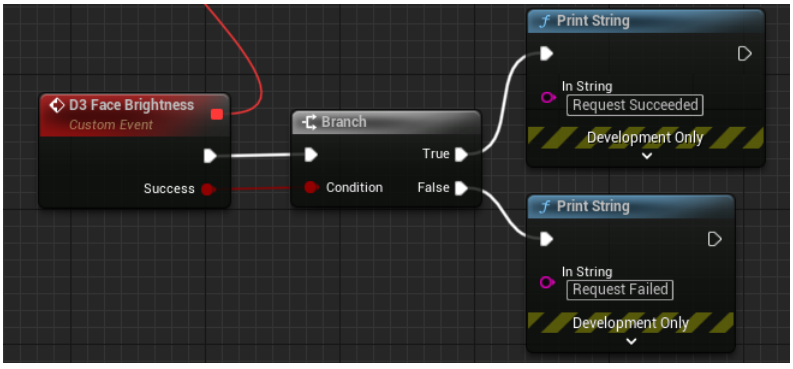
      The **Output** pin of the **Custom Event** node is automatically connected to the **Input** pin of the **Branch** node.

   d. In the **Branch** node, left-click and drag off from the **True** pin, begin typing `print string` and select the **Print String** node.

   e. In the **Print String** node, in the **In String** field, enter some text to indicate success (e.g., Request Succeeded).

   f. In the **Branch** node, left-click and drag off from the **False** pin, begin typing `print string` and select the **Print String** node.

   g. In the **Print String** node, in the **In String** field, enter some text to indicate failure (e.g., Request Failed).



D3 Feedback Nodes

When the command is executed, the success or failure of the command is indicated.

9. Continue with the procedure for the specific D3 blueprint you want.

   Get D3 Face Settings

   Set D3 Face Settings

   Get D3 Faces Id List

   Set D3 Face Brightness

   Set D3 Face Offsets

   Set D3 Face Color Adjustments

To create a Get D3 Face Settings blueprint:

1. In the **Custom Event** node, right-click the **D3 Face Settings** pin and from the menu, select **Split Struct Pin**.

   The **D3 Face Settings** pin is expanded to show all the settings.

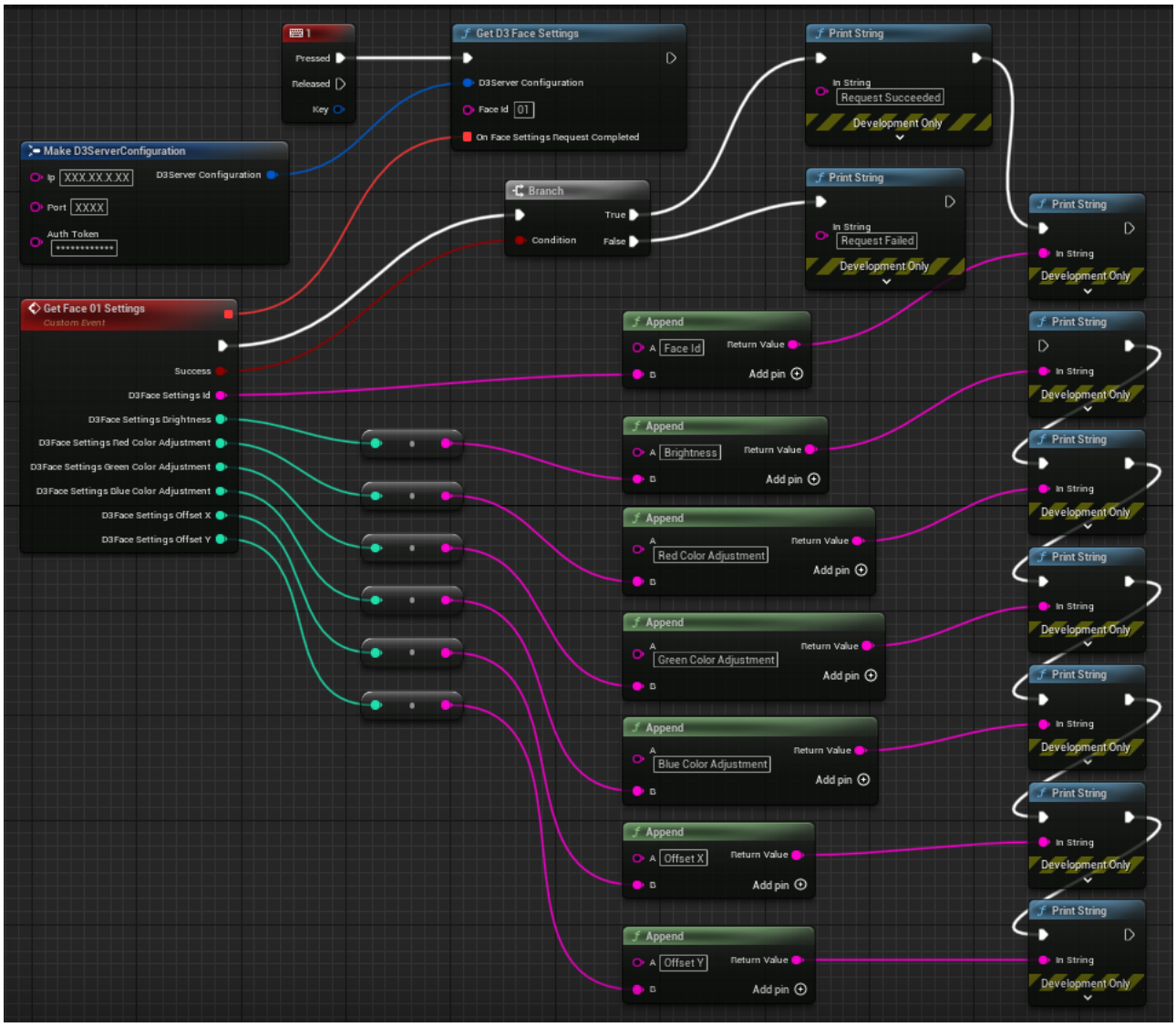
2. Left-click and drag off the **D3 Face Settings Id** pin, begin typing `string append` and select the **Append** node.
3. Disconnect the **D3 Face Settings Id** pin from the **A Input** pin of the **Append** node and connect it instead to the **B Input** pin.
4. Then in the **A Input** field of the **Append** node, enter the name of the setting (e.g., Face Settings ID).
5. For any of the remaining **D3 Face Settings**, left-click and drag off from the pin, begin typing `string append` and select the **Append** node.

   The **To String (Integer)** node is automatically inserted between the setting pin and the **Append** node.

6. Disconnect the setting pin from the **A Input** pin of the **Append** node and connect it instead to the **B Input** pin.
7. In each **Append** node, in the **A Input** field, enter the name of the setting from which it originates (e.g., Brightness, Red Color Adjustment, etc.)
8. For each **Append** node, left-click and drag off from the **Return Value** pin, begin typing `print string` and then select the **Print String** node.

9. Connect the **Output** pin of one **Print String** node to the **Input** pin of the next **Print String** node.

   Your blueprint will be similar to the blueprint below:

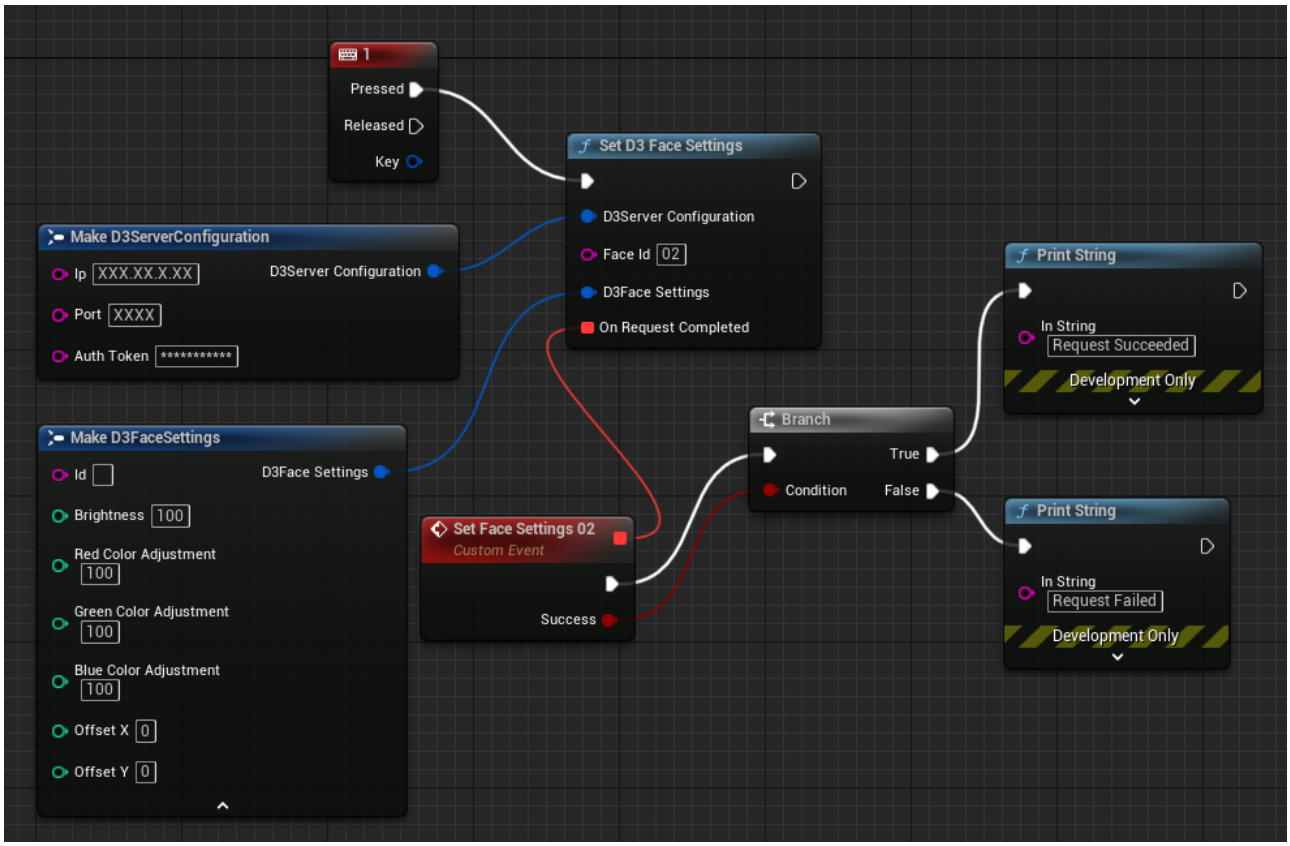


Get D3 Face Settings Blueprint

To create a Set D3 Face Settings blueprint:

1. In the **Set D3 Face Settings** node, left-click and drag off from the **D3Face Settings** pin and select the **Make D3FaceSettings** node.
2. In the **Make D3FaceSettings** node, set the required parameters as described below:
   - In the **Id** field, enter the identification number of the display you want to configure.
   - In the **Brightness** field, enter a value from **0 - 100** for the level of brightness for the display.
   - In the **Red**, **Green** and **Blue Color Adjustment** fields, enter values from **0 - 100** to modify those colors in the display.
   - In the **Offset X** field, enter the pixel value from the top left of the screen to offset the input in the display, on the X axis.
   - In the **Offset Y** field, enter the pixel value from the top left of the screen to offset the input in the display, on the Y axis.

   Your blueprint will be similar to the blueprint below:

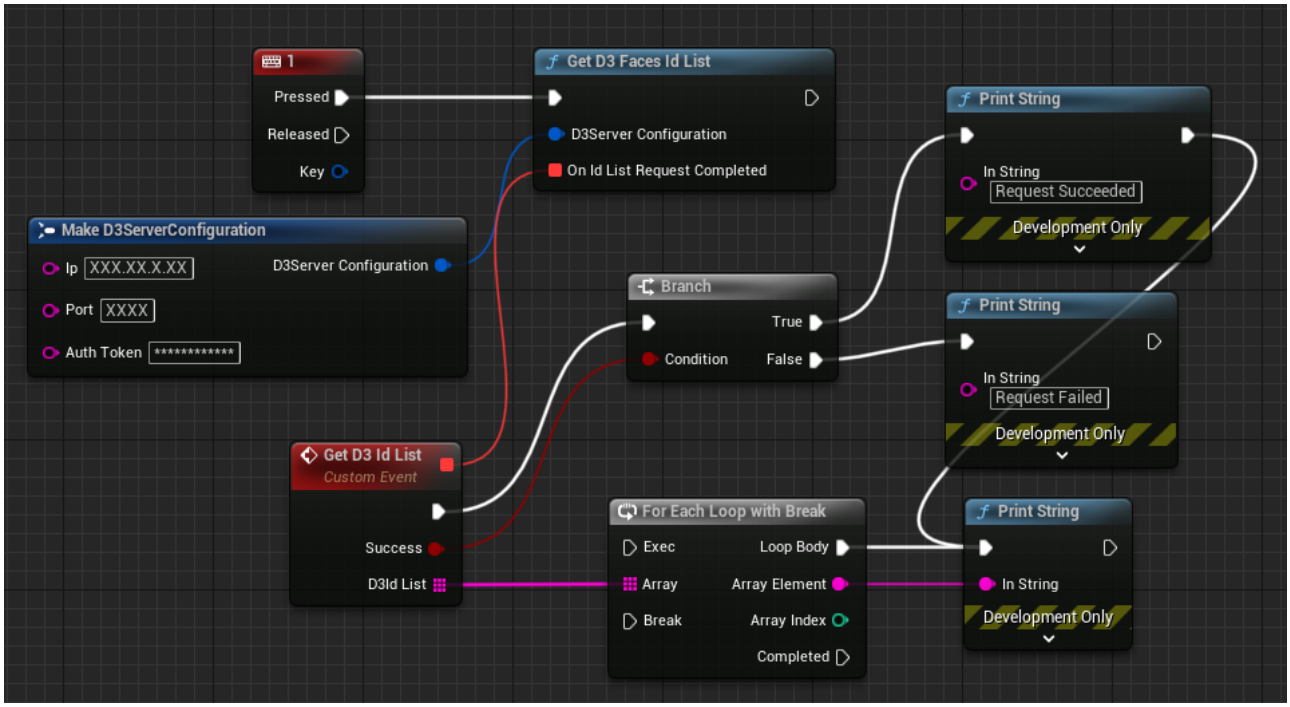


Set D3 Face Settings Blueprint

To create a Get D3 Faces Id List blueprint:

1. In the **Custom Event** node, left-click and drag off the **D3Id List** pin, begin typing `loop` and select the **For Each Loop with Break** node.
2. In the **For Each Loop with Break** node, left-click and drag off the **Array Element** pin, begin typing `print string` and select the **Print String** node.
3. Connect the **Loop Body** pin to the **Print String Input** pin.
4. Connect the **Output** pin of the **Print String** node that is connected to the **True** pin of the **Branch** node to the **Input** pin of the new **Print String** node.

   Your blueprint will be similar to the blueprint below:

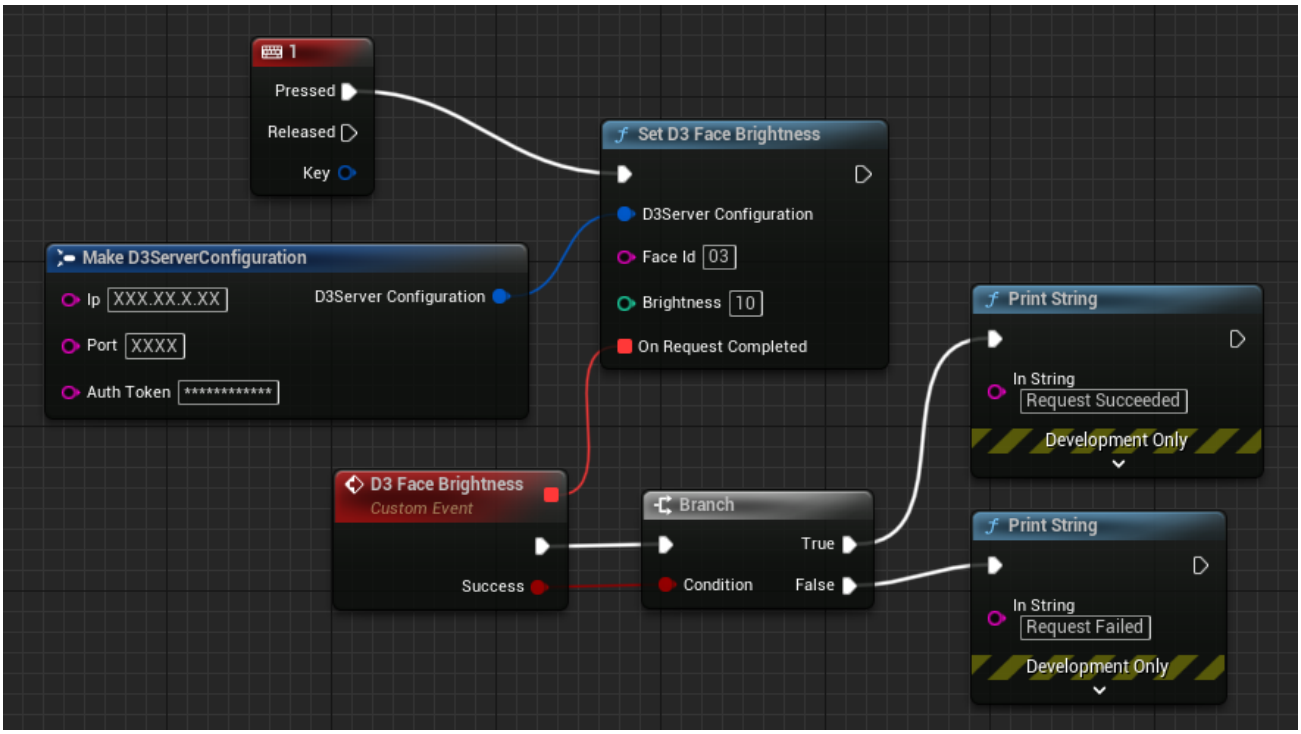


Get D3 Faces Id List Blueprint

To create a Set D3 Face Brightness blueprint:

- In the **Set D3 Face Brightness** node, in the **Brightness** field, enter a value from **0 - 100** for the level of brightness of the selected face.

  Your blueprint will be similar to the blueprint below:

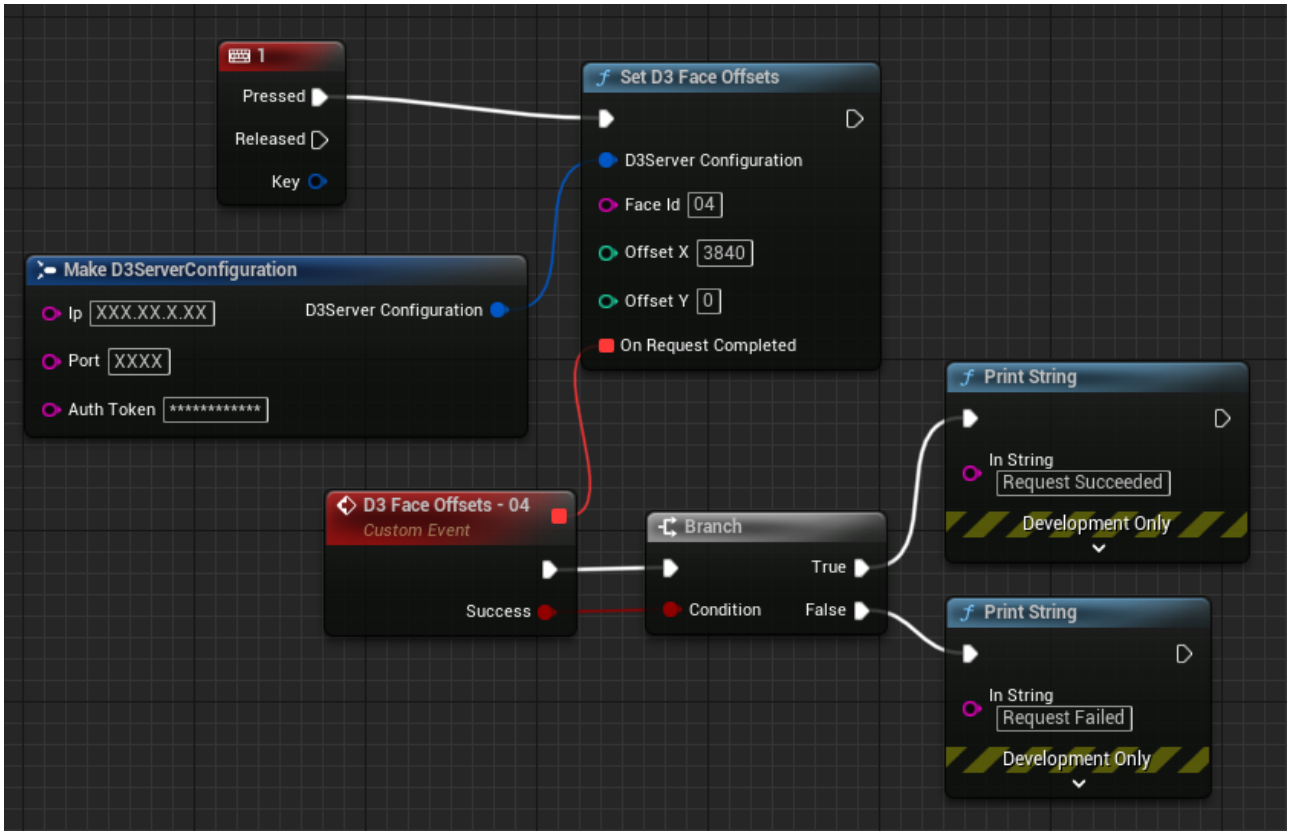


Set D3 Face Brightness Blueprint

To create a Set D3 Face Offsets blueprint:

1. In the **Set D3 Face Offsets** node, in the **Offset X** field, enter a pixel value from the top-left of the screen for the amount of offset required for the selected face, on the **X** axis.
2. In the **Offset Y** field, enter a pixel value from the top-left of the screen for the amount of offset required for the selected face, on the **Y** axis.

   Your blueprint will be similar to the blueprint below:

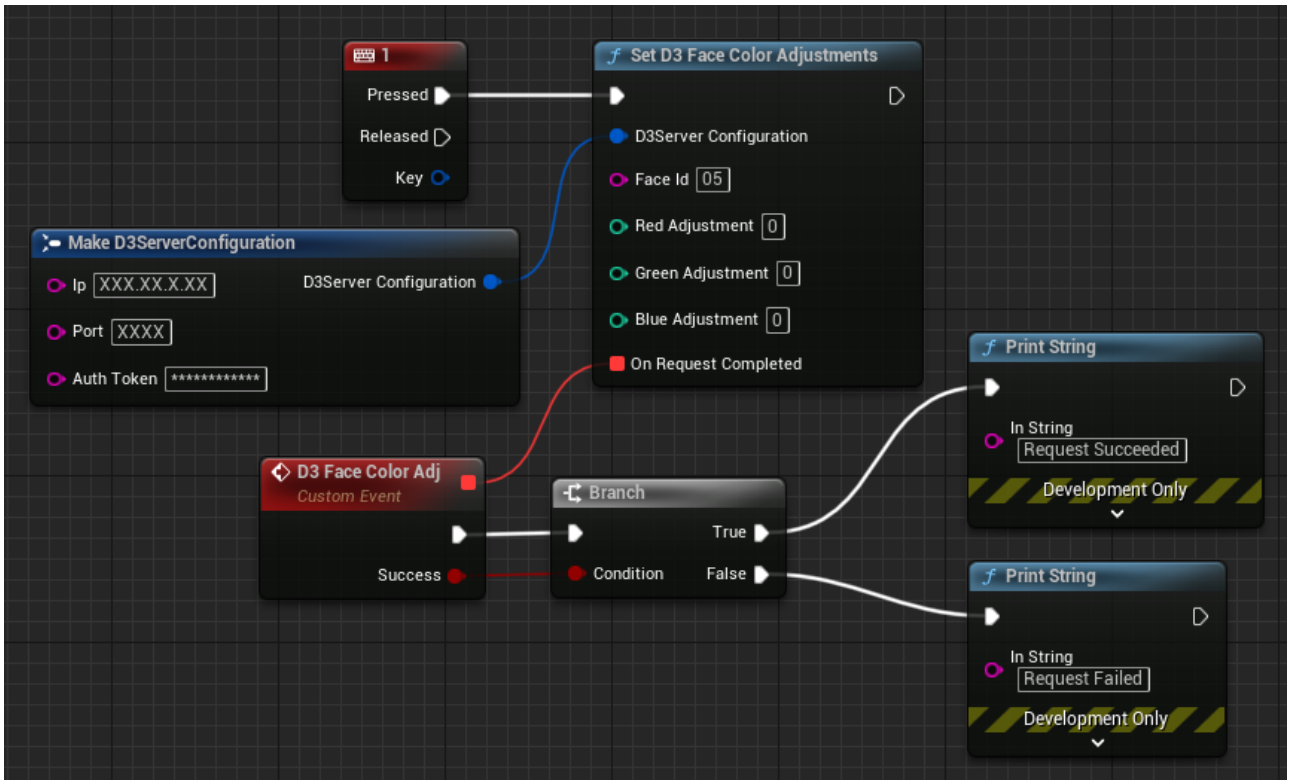


Set D3 Face Offsets Blueprint

To create a Set D3 Face Color Adjustments blueprint:

- In the **Set D3 Face Color Adjustments** node, enter values from **0 - 100** in the **Red Adjustment**, **Green Adjustment** and **Blue Adjustment** fields to modify those colors in the selected face.

  Your blueprint will be similar to the blueprint below:



Set D3 Face Color Adjustments Blueprint

Playlist Commands

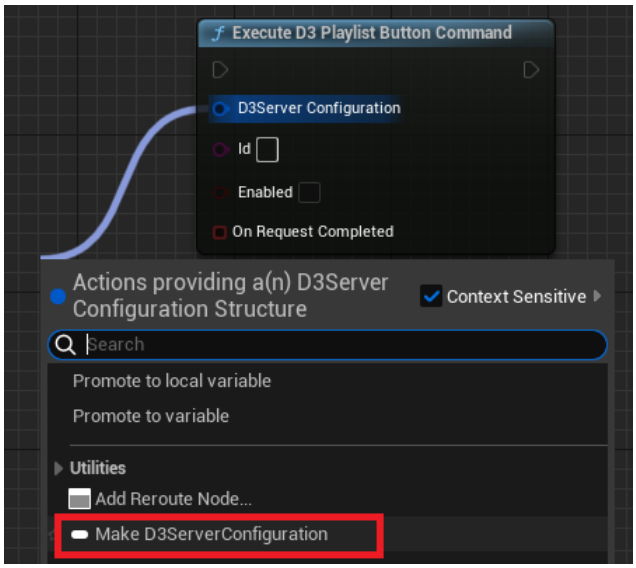
Use the Playlist Commands to control the playout of videos and to select the playlist you want to use. There are 2 playlist commands:

Execute D3 Playlist Button Command — switches between playlists

Execute D3 Playlist Command — controls the playout of a video playlist (Play, Pause, Next, Previous and Stop)

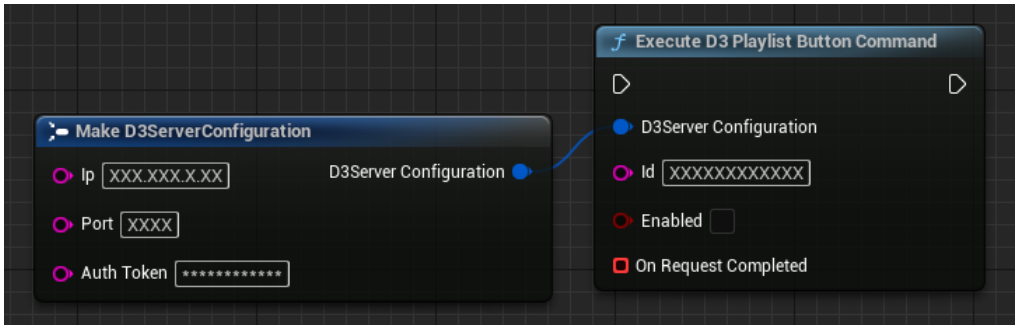
To use the Execute D3 Playlist Button Command:

1. Select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.
2. Right-click in the blueprint and in the **Search** field, begin typing `D3` and from the results, select the **Execute D3 Playlist Button Command** node.
3. Left-click and drag off from the **D3Server Configuration** pin and from the **Actions providing a(n) D3Server Configuration Structure** menu, select **Make 3DServerConfiguration**.



Make D3ServerConfiguration - Playlist Button

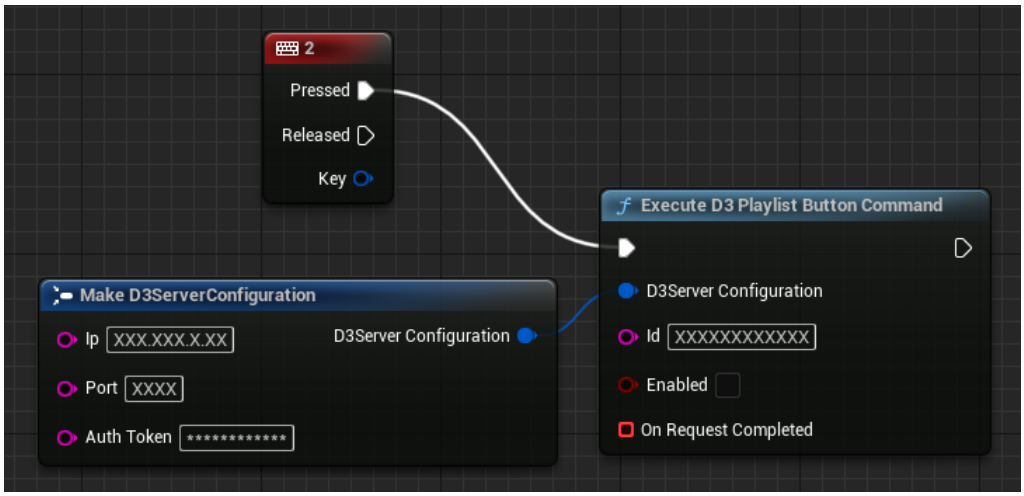
4. In the **Make D3ServerConfiguration** node, do the following:
   - In the **Ip** field, enter the IP address of the D3 server.
   - In the **Port** field, enter the port number for the D3 server.
   - In the **Auth Token** field, enter your token number.



D3 Server Configuration - Playlist Button

5. Right-click in the blueprint and select a trigger (e.g., a button press, or a RossTalk GPI Event node).
6. Connect the **Output** pin of the trigger node to the **Input** pin of the **Execute D3 Playlist Button Command** node.
7. In the **Execute D3 Playlist Button Command** node, enter the number of the Playlist Button.

   Your blueprint will be similar to the blueprint below:



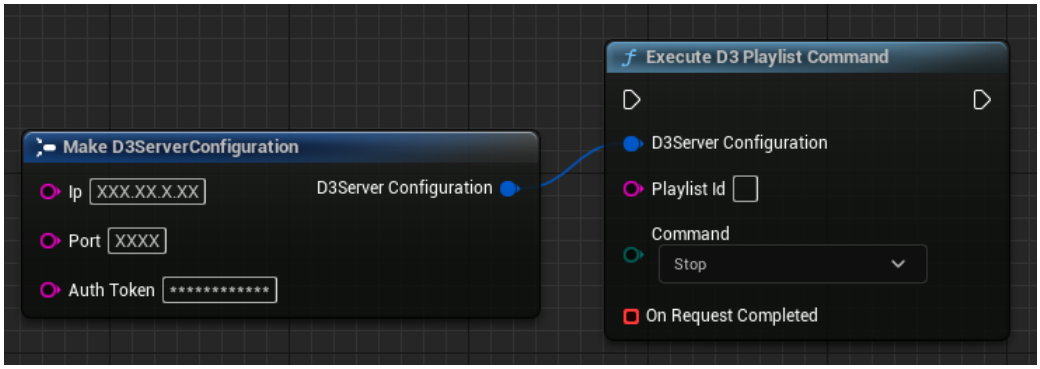
Execute D3 Playlist Button Command Blueprint

To use the Execute D3 Playlist Command node:

1. Select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.
2. Right-click in the blueprint and in the **Search** field, begin typing `D3` and from the results, select the **Execute D3 Playlist Command** node.
3. Left-click and drag off from the **D3Server Configuration** pin and from the **Actions providing a(n) D3Server Configuration Structure** menu, select **Make 3DServerConfiguration**.

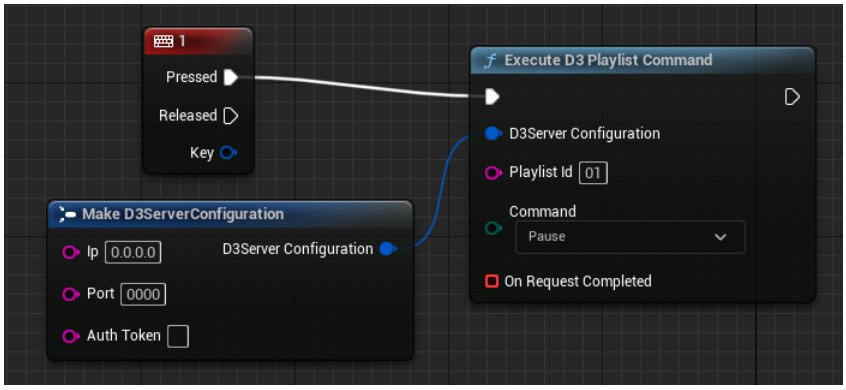
Make D3ServerConfiguration - Playlist

4. In the **Make D3ServerConfiguration** node, do the following:
   - In the **Ip** field, enter the IP address of the D3 server.
   - In the **Port** field, enter the port number for the D3 server.
   - In the **Auth Token** field, enter your token number.



D3 Server Configuration - Playlist

5. Right-click in the blueprint and select a trigger (e.g., a button press, or a RossTalk GPI Event node).
6. Connect the **Output** pin of the trigger node to the **Input** pin of the **Execute D3 Playlist Command** node.



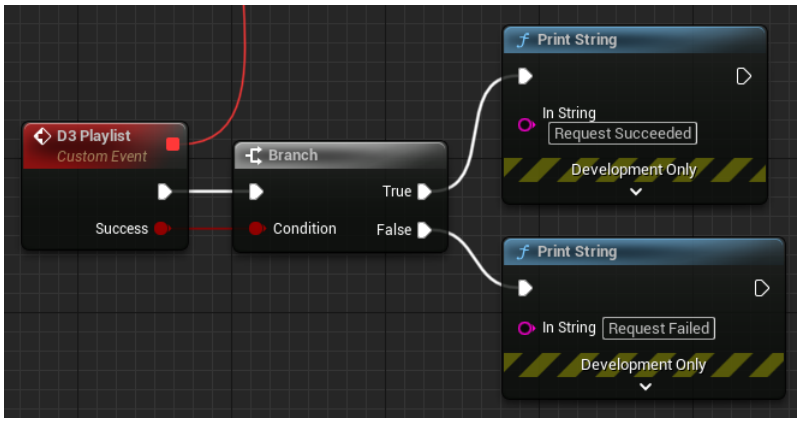
Add Trigger

7. Add **Feedback** nodes as follows:
   a. In the **Execute D3 Playlist Command** node, left-click and drag off from the **On Request Completed** pin, begin typing add custom event and then select the **Add Custom Event** node.
   b. Give the **Custom Event** node a unique name that corresponds to the **Execute D3 Playlist Command** node.
   c. In the **Custom Event** node, left-click and drag off from the **Success** pin, begin typing `branch` and then select the **Branch** node.

      The **Output** pin of the **Custom Event** node is automatically connected to the **Input** pin of the **Branch** node.
   d. In the **Branch** node, left-click and drag off from the **True** pin, begin typing `print string` and then select the **Print String** node.
   e. In the **Print String node**, in the **In String** field, enter some text to indicate success (e.g., Request Succeeded).

f. In the **Branch** node, left-click and drag off from the **False** pin, begin typing `print string` and then select the **Print String** node.

g. In the **Print String** node, in the **In String** field, enter some text to indicate failure (e.g., Request Failed).



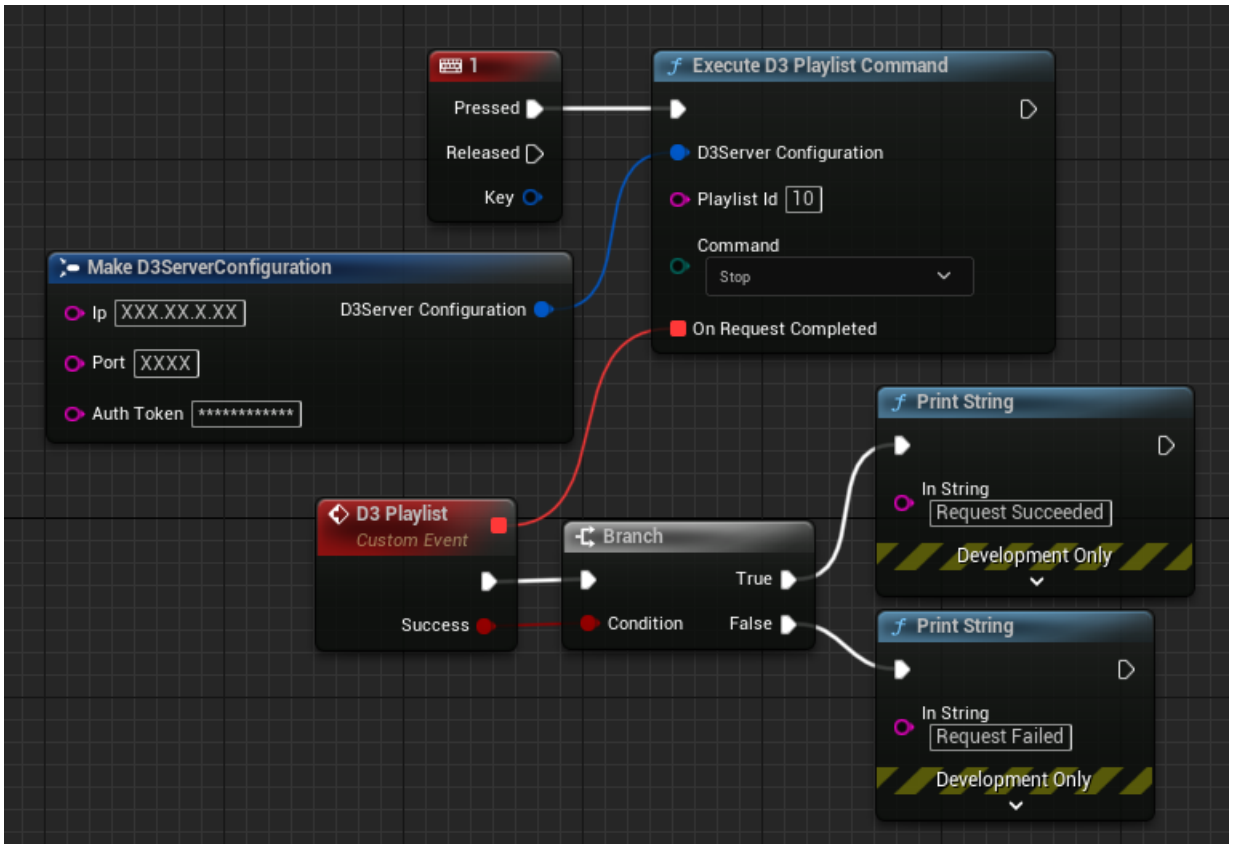
D3 Feedback Nodes

When the command is executed, the success or failure of the command is indicated.

8. In the **Execute D3 Playlist Command** node, do the following:

- In the **Playlist Id** field, enter the number of the playlist you want to control.
- From the **Command** drop-down, select the action you want to initiate when the trigger is received, either **Stop**, **Previous**, **Pause**, **Play**, or **Next**.

Your blueprint will be similar to the blueprint below:



Execute D3 Playlist Command Blueprint

Voyager Node

The Voyager Node allows users to run Voyager projects in **Game** mode only (no **Editor** user interface). Upon installation, a new desktop shortcut is created, which enables the user to launch Voyager Node and select a project to run in game mode.

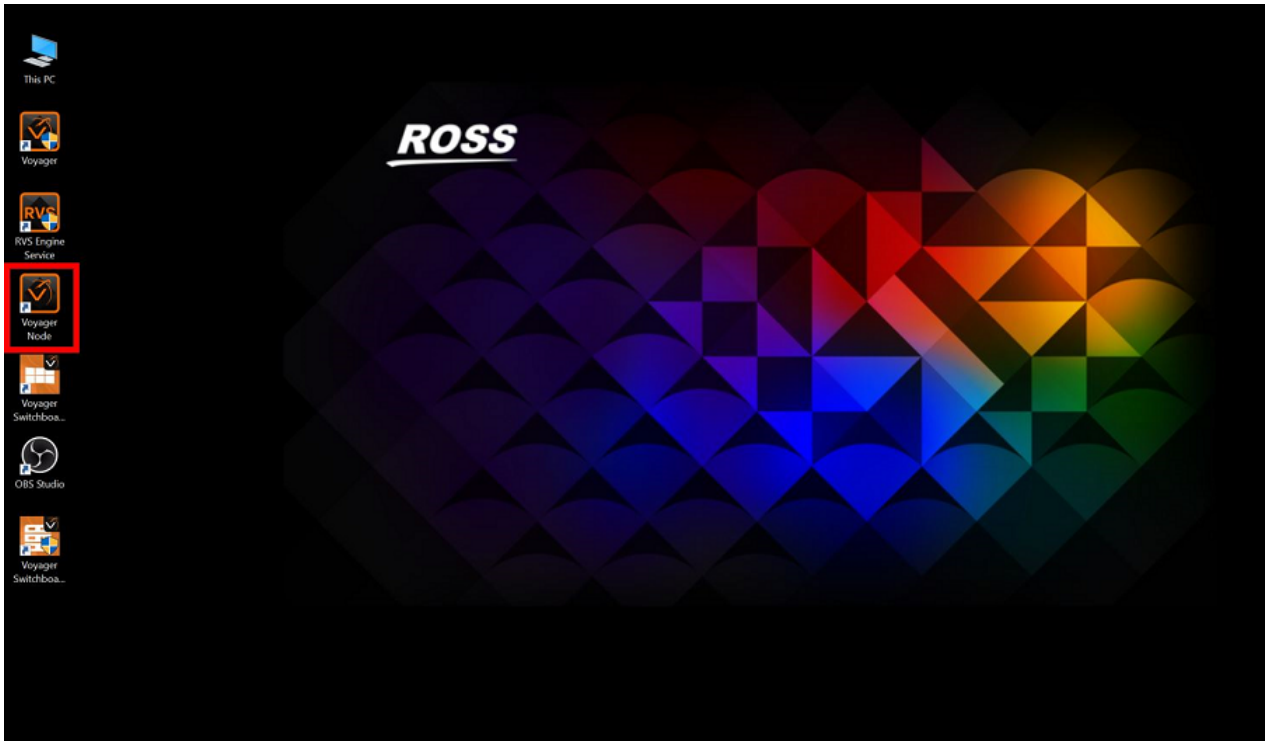
A full Voyager license or a Voyager Node license is required to run Voyager Node.

Additionally, users can run XR/nDisplay nodes if they have a Voyager Node License. As XR nodes operate in GAME mode when running an XR/nDisplay project, you do not require a full Voyager license. However, you will need the following licenses to run the XR nodes:

- **nDisplay Controller** — use for the main XR node.
- **nDisplay Node** — use for any secondary XR node.

To run Voyager Node:

1. On the desktop, double-click the **Voyager Node** shortcut.



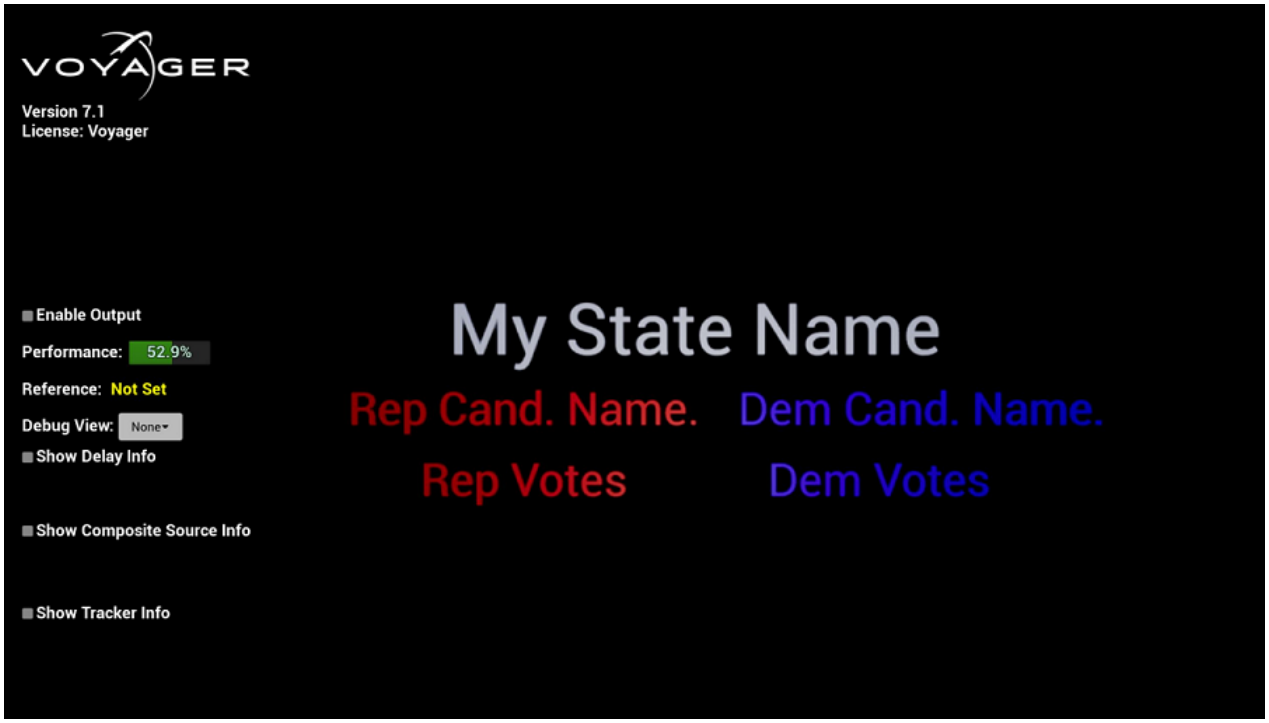
Voyager Node Desktop Shortcut

A project explorer window opens.

2. Navigate to the project you want and select **Open**.

   Voyager starts in **Game** mode and runs the selected project.

   ★When running in **Game** mode, the project's **Media Profile** needs to be set as the **Startup Media Profile**. See Selecting the Startup Media Profile for instructions.



Project Running in Game Mode

To close Voyager Node:

- On the keyboard, press **SHIFT**+**ESC**.

  The **Voyager Node** closes.

Using RossTalk

You can use RossTalk commands from a switcher or control application to trigger events in Voyager.

To use RossTalk in Voyager, you need to have the RossTalk Plugin enabled. The plugin is enabled by default.

You'll also need to configure the RossTalk settings. Once the settings are configured, you can use the following RossTalk nodes in your blueprints:

On RossTalk CC Event

On RossTalk NEXT Event

On RossTalk GPI Event

Send RossTalk CC

Send RossTalk Next

Send RossTalk GPI

Send RossTalk SEQI

Send RossTalk Custom Command

To print the values and states of any of the GPI settings, see To display the GPI values and states.

To initialize multiple RossTalk events, see To initialize multiple RossTalk events.

To configure the RossTalk Settings:

1. From the **Window** menu, select **Voyager > RossTalk Settings**.
2. From the **Listening IP/Network** drop-down, select the network device on which to listen to RossTalk commands.

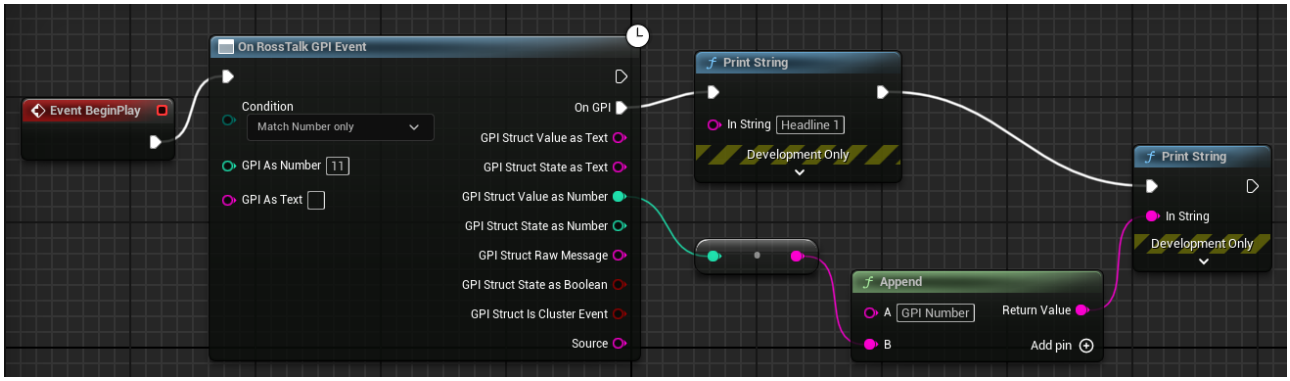
   You can select a specific device or select **Any IP** to allow multiple devices to listen to RossTalk commands.
3. In the **Listening Port** field, enter the number of the port on which the device(s) will listen for RossTalk commands.

   The default port is **7799**.
4. Then select **Apply changes and restart RossTalk Server**.

To display the GPI values and states:

1. Right-click the **GPI Struct** pin and select **Split Struct Pin**.
2. For each setting you want to get the value or state of, right-click in an empty part of the blueprint, begin typing `string append` and select the **Append** node.
3. For pins that are integers, insert a **To String (Integer)** and for pins that are **Booleans**, insert a **To String (Boolean)** node between the pin and the **Append** node.
4. Connect the pin or the **Output** pin of the **To String (Integer)** or **To String (Boolean)** node to the **B Input** pin of the **Append** node.
5. In the **Append** node, in the **A Input** field, enter the name of the pin.

6. Left-click and drag off the **Return Value** pin of the **Append** node, begin typing `print string` and select the **Print String** node.
7. Connect the **Output** pin of the node that is being triggered to the **Input** pin of the **Print String** node.
8. If you are getting more than one value, connect the **Output** pin of the first **Print String** node to the **Input** pin of the next **Print String** node, until all the nodes are connected.

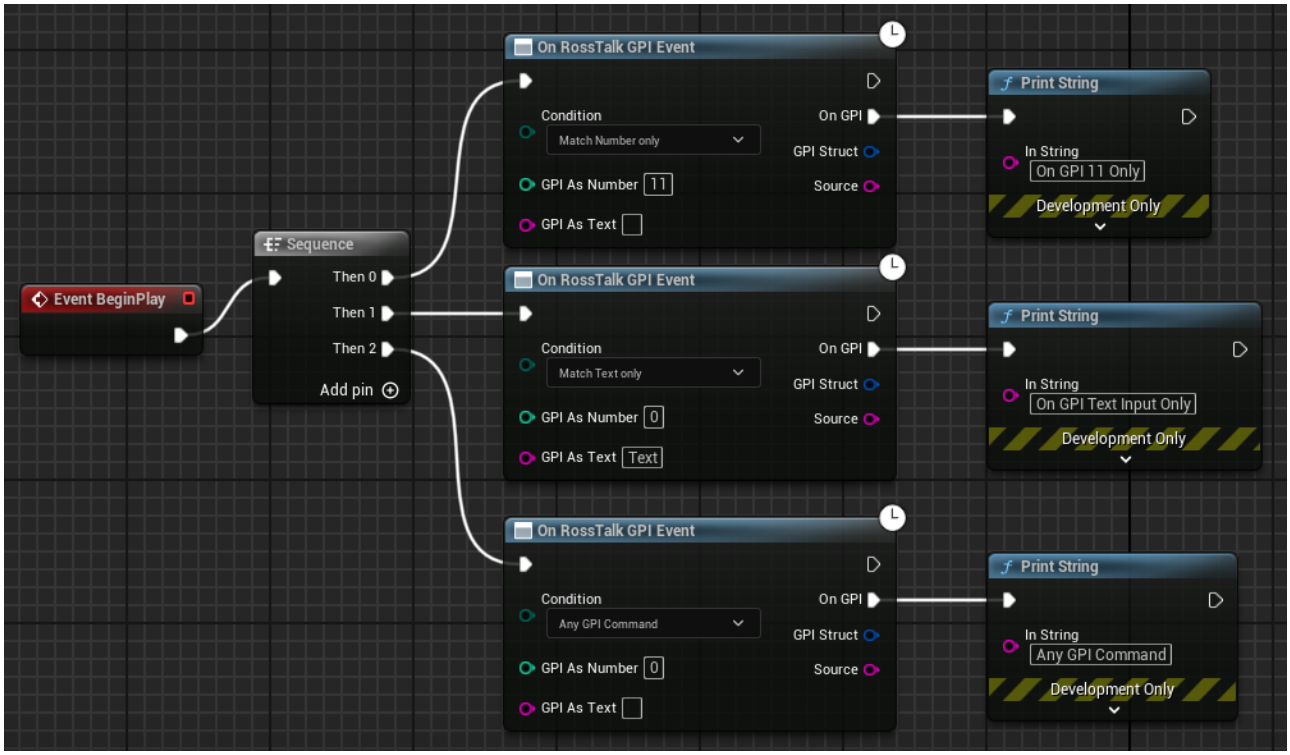


On RossTalk GPI Event with Struct

To initialize multiple RossTalk events:

1. In any of the blueprints created in the above procedures, right-click in the blueprint and in the **Search** field, begin typing `Sequence` and select the **Sequence** node.
2. Break the link between the **Event BeginPlay** node and the RossTalk node and instead, connect the **Event BeginPlay** node **Output** pin to the **Input** pin of the **Sequence** node.
3. Then connect the **Output** pin of the **Sequence** node to the **Input** pin of the RossTalk node.
4. Add as many RossTalk nodes as you require and connect them to the **Sequence** node as described in the previous step.
5. If you require more than 2 RossTalk nodes, select **Add pin** on the **Sequence** node to add a pin for each RossTalk node.

   In the example below, all of the events will be executed when any GPI command is received.



Initializing Multiple RossTalk Events

On RossTalk CC Event

A **RossTalk CC Event** is triggered when an incoming RossTalk CC command is received.

To use an On RossTalk CC Event node:

1. Select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.
2. Right-click in the blueprint and in the **Search** field, begin typing `Event Begin Play` and then select the **Event Begin Play** node.
3. Right-click in the blueprint and in the **Search** field, begin typing `RossTalk`.
4. From the search results, select the **On RossTalk CC Event** node.
5. Connect the **Output** pin of the **Event BeginPlay** node to the **Exec** input pin of the **On RossTalk CC Event** node.
6. In the **On RossTalk CC Event** node, do the following:
   - From the **Condition** drop-down, select whether to match **Any Custom Control Command**, **Match Bank and CC**, **Match CC Only**, or **Match Bank** Only.
   - If you selected **Match Bank and CC**, in the **Bank** field enter the number of the bank (a row on a switcher) and in the **CC** field, enter the number of the custom control.
   - If you selected **Match CC Only**, in the **CC** field, enter the number of the custom control command that must be matched.
   - If you selected **Match Bank Only**, in the **Bank** field, enter the number of the bank that must be matched.
7. Then connect the **Exec** output pin of the **On RossTalk CC Event** node to the **Exec** input pin of the event you want to trigger.
8. Save and compile the blueprint.

On RossTalk NEXT Event

An **On RossTalk NEXT Event** is triggered when an incoming RossTalk NEXT command is received.

To use an On RossTalk NEXT Event node:

1. Select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.
2. Right-click in the blueprint and in the **Search** field, begin typing `Event BeginPlay` and then select the **Event BeginPlay** node.
3. Right-click in the blueprint and in the **Search** field, begin typing `RossTalk`.
4. From the search results, select the **On RossTalk NEXT Event** node.
5. Connect the **Output** pin of the **Event BeginPlay** node to the **Exec** input pin of the **On RossTalk Next Event** node.
6. Connect the **On Next** pin of the **On RossTalk Next Event** node to the **Exec** input pin of the event you want to trigger.
7. Save and compile the blueprint.

On RossTalk GPI Event

An **On RossTalk GPI** Event is triggered when an incoming RossTalk GPI command is received.

To use an On RossTalk GPI Event node:

1. Select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.
2. Right-click in the blueprint and in the **Search** field, begin typing `Event BeginPlay` and then select the **Event BeginPlay** node.
3. Right-click in the blueprint and in the **Search** field, begin typing `RossTalk`.
4. From the search results, select the **On RossTalk GPI Event** node.
5. Connect the **Output** pin of the **Event BeginPlay** node to the **Exec** input pin of the **RossTalk GPI Event** node.
6. In the **On RossTalk GPI Event** node, do the following:
   - From the **Condition** drop-down, select whether to match **Any GPI Command**, **Match Number only**, **Match Text only** or **Match Number or Text**.
   - If you selected **Match Number only** or **Match Number or Text**, in the **GPI as Number** field, enter the number of the GPI command that must be matched.
   - If you selected **Match Text only** or **Match Number or Text**, in the **GPI as Text** field, enter the text of the GPI command that must be matched.
7. Then connect the **Exec** output pin of the **On RossTalk GPI Event** node to the **Exec** input pin of the event you want to trigger.

   In the example below, the **On RossTalk GPI Event** is triggering the printing of a headline to the screen.

On RossTalk GPI Event

8. Save and compile the blueprint.

Send RossTalk CC

Use the **Send RossTalk CC** node to send a **RossTalk Custom Control Command** to other Ross equipment.

To use a Send RossTalk CC node:

1. Select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.
2. Right-click in the blueprint and in the **Search** field, begin typing `RossTalk`.
3. From the search results, select the **Send RossTalk CC** node.
4. Add an input node (e.g., Lucid Exec, Voyager Event Execution, Key Press, etc.) and connect the **Output** pin of the input node to the **Exec** input pin of the **Send RossTalk CC** node.
5. In the **Send RossTalk CC** node, do the following:
   - In the **IP** field, enter the IP address of the Ross equipment.
   - In the **Port** field, enter the Port number on which Voyager will communicate with the equipment.
   - In the **Value** field, enter the number of the custom control command to send.
6. Save and compile the blueprint.

Send RossTalk Next

Use the **Send RossTalk Next** node to send a **RossTalk Next** command to other Ross equipment.

To use a Send RossTalk Next node:

1. Select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.
2. Right-click in the blueprint and in the **Search** field, begin typing `RossTalk`.
3. From the search results, select the **Send RossTalk Next** node.
4. Add an input node (e.g., Lucid Exec, Voyager Event Execution, Key Press, etc.) and connect the **Output** pin of the input node to the **Exec** input pin of the **Send RossTalk Next** node.
5. In the **Send RossTalk Next** node, do the following:
   - In the **IP** field, enter the IP address of the Ross equipment.
   - In the **Port** field, enter the Port number on which Voyager will communicate with the equipment.
6. Save and compile the blueprint.

Send RossTalk GPI

Use the **Send RossTalk GPI** node to send a **RossTalk GPI** command to other Ross equipment.

To use a Send RossTalk GPI node:

1. Select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.
2. Right-click in the blueprint and in the **Search** field, begin typing `RossTalk`.
3. From the search results, select the **Send RossTalk GPI** node.
4. Add an input node (e.g., Lucid Exec, Voyager Event Execution, Key Press, etc.) and connect the **Output** pin of the input node to the **Exec** input pin of the **Send RossTalk GPI** node.
5. In the **Send RossTalk GPI** node, do the following:
   - In the **IP** field, enter the IP address of the Ross equipment.
   - In the **Port** field, enter the Port number on which Voyager will communicate with the equipment.
   - In the **Value** field, enter the number of the GPI command to send.
6. Save and compile the blueprint.

Send RossTalk SEQI

Use the **Send RossTalk SEQI** node to send a **RossTalk SEQI** command to other Ross equipment.

To use a Send RossTalk SEQI node:

1. Select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.
2. Right-click in the blueprint and in the **Search** field, begin typing `RossTalk`.
3. From the search results, select the **Send RossTalk SEQI** node.
4. Add an input node (e.g., Lucid Exec, Voyager Event Execution, Key Press, etc.) and connect the **Output** pin of the input node to the **Exec** input pin of the **Send RossTalk SEQI** node.
5. In the **Send RossTalk SEQI** node, do the following:
   - In the **IP** field, enter the IP address of the Ross equipment.
   - In the **Port** field, enter the Port number on which Voyager will communicate with the equipment.
   - In the **Value** field, enter the **Take ID** of the scene to send to air.
6. Save and compile the blueprint.

Send RossTalk Custom Command

Use the **Send RossTalk Custom Command** node to send a **Custom RossTalk Command** to other Ross equipment.

To use a Send RossTalk Custom Command node:

1. Select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.
2. Right-click in the blueprint and in the **Search** field, begin typing `RossTalk`.
3. From the search results, select the **Send RossTalk Custom Command** node.
4. Add an input node (e.g., Lucid Exec, Voyager Event Execution, Key Press, etc.) and connect the **Output** pin of the input node to the **Exec** input pin of the **Send RossTalk Custom Command** node.
5. In the **Send RossTalk Custom Command** node, do the following:
   - In the **IP** field, enter the IP address of the Ross equipment.
   - In the **Port** field, enter the Port number on which Voyager will communicate with the equipment.
   - In the **Command** field, enter the command you want to send.
6. Save and compile the blueprint.

Using the Voyager Web API

The Voyager Web API allows an authorized user to access OpenAPI documentation to retrieve detailed information on the processes and to understand and interact with remote engines without needing to log into each engine. The documentation provides a complete list of endpoints that can be used for third party integration and testing. For a more user-friendly UI, you can use the Swagger version.

The machine from which you are accessing the Web API must be on the same network or VPN as the Voyager engines.

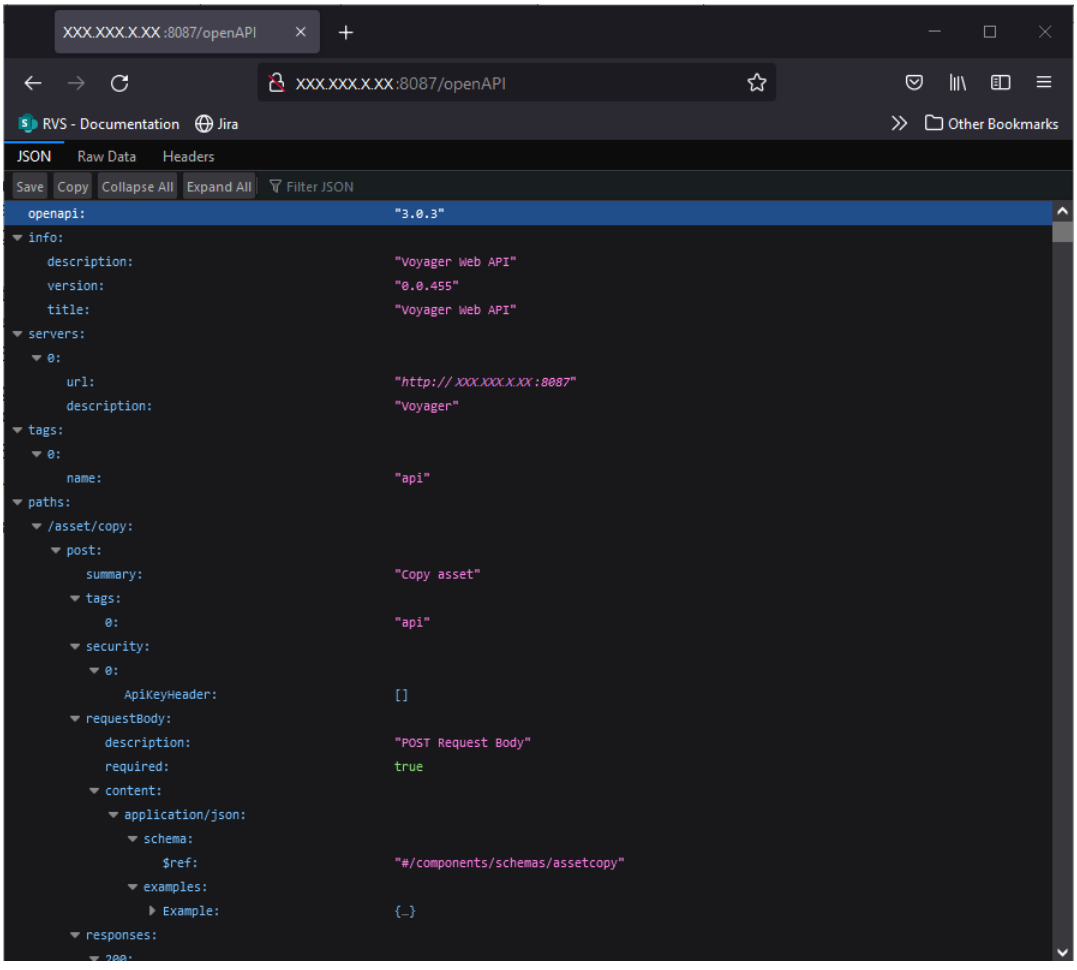
For instruction on enabling the Voyager Web API Plugin, see Enabling the Voyager Plugins.

You can increase the security of your Voyager network by whitelisting the machines that can access the Web API. See Whitelisting IP Addresses for instructions.

OpenAPI

The OpenAPI documentation can be accessed at a URL defined by the IP address of the Voyager engine followed by the port number and "openAPI" (eg. http://xx.xx.xxx.xx:8087/openAPI).

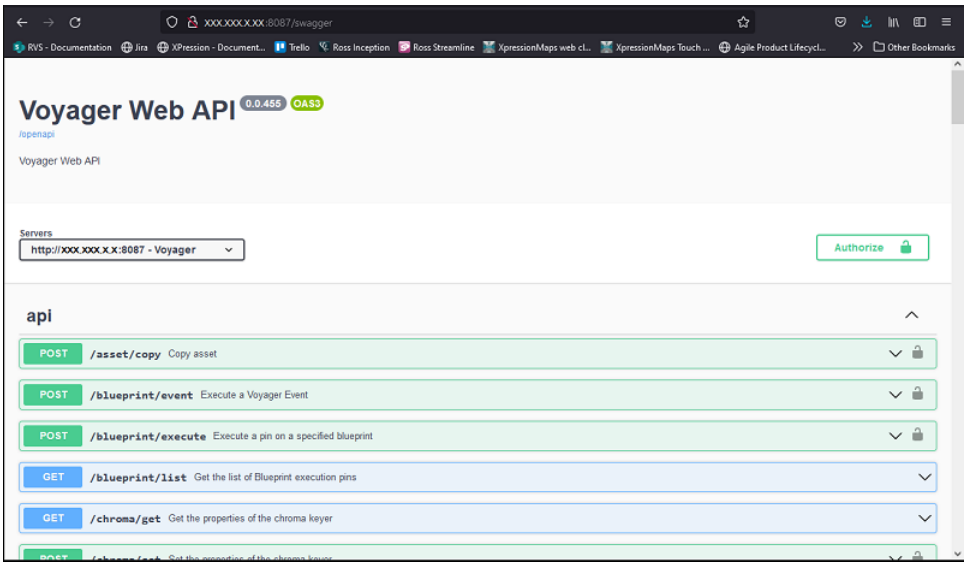
It provides a complete list of endpoints that can be used for third party integration.



Voyager Web API - OpenAPI Documentation

Swagger

Swagger is the more user-friendly version of the API. It can be accessed using a URL defined by the IP address of the Voyager engine followed by the port number and "swagger" (eg. http://xx.xx.xxx.xx:8087/swagger).

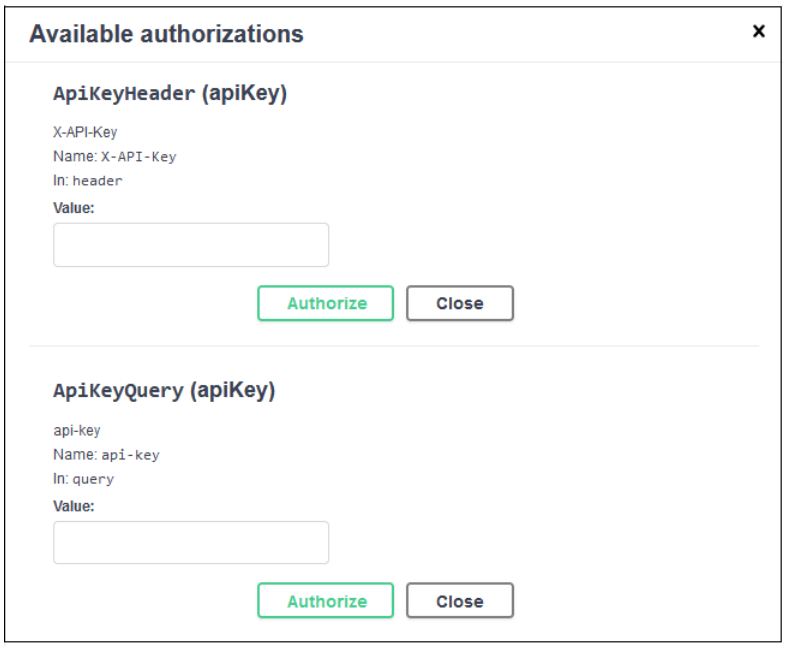


Voyager Web API - Swagger Documentation

To use Swagger:

1. Select the **Authorize** button.

   The **Available authorizations** window opens.



   Voyager Web API - Swagger Authorization

2. In the **ApiKeyHeader (apiKey)** section, enter your **Web API** key in the **Value** field and select **Authorize**.

Whitelisting IP Addresses

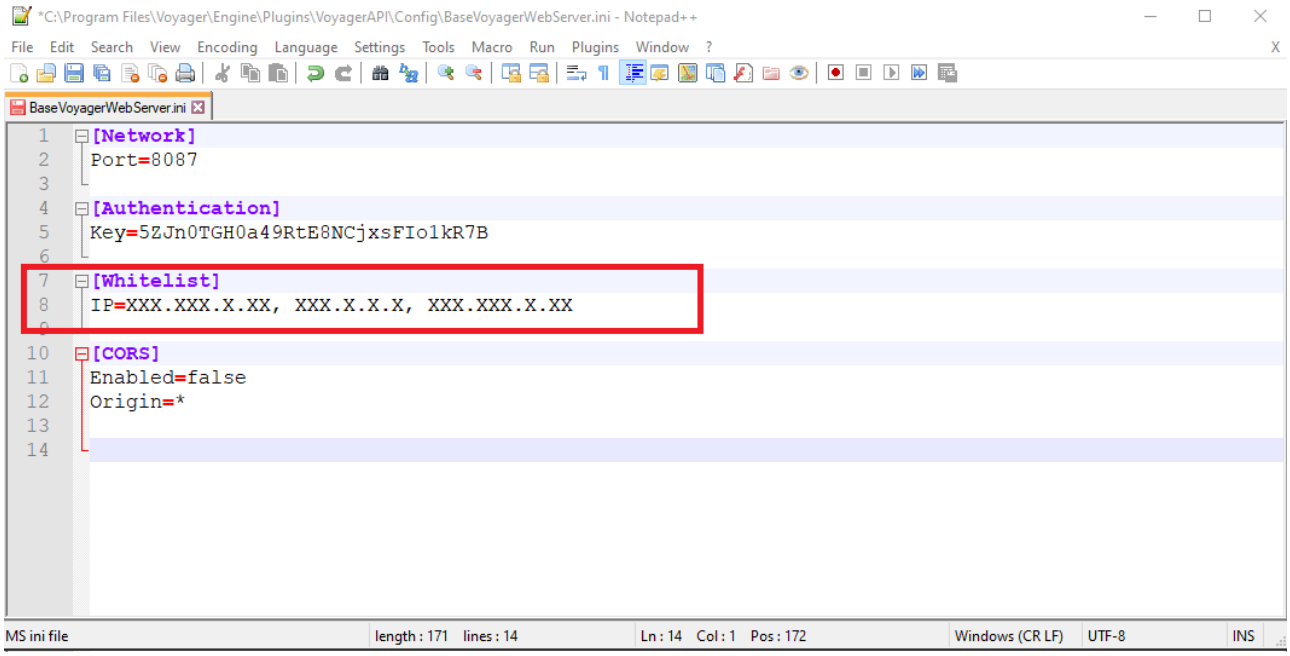
You can restrict access to the Voyager Web API to a select group of machines by whitelisting their IP addresses.This provides greater control and security.

To whitelist IP addresses:

1. Navigate to the Voyager configuration file, typically located at the address below:

   C:\Program Files\Voyager\Engine\Plugins\VoyagerAPI\Config.

2. Open the **Config** file with a text editor.
3. Add the list of IP addresses, separated by a comma, as shown in the image below:



Voyager Web API - Add Whitelist

4. Select the **Save** button.

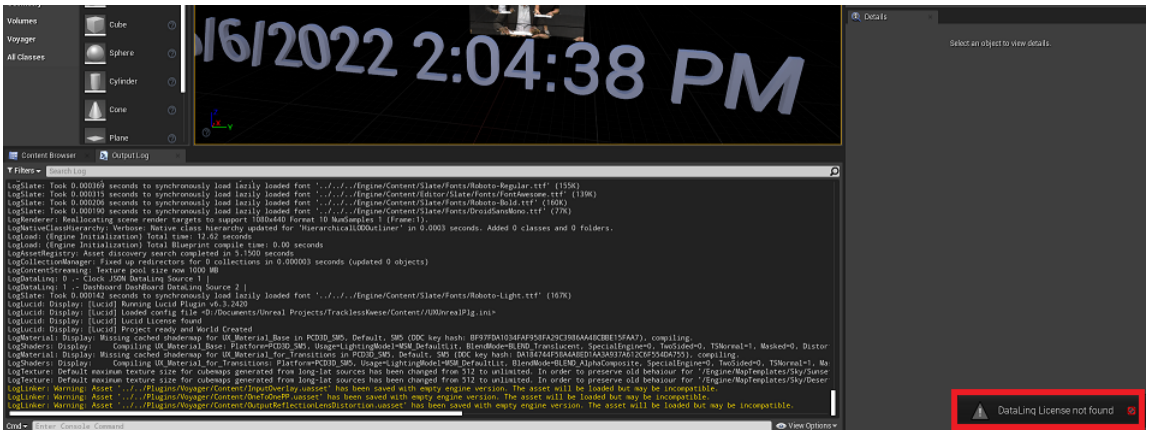
Using DataLinq in Your Project

With the DataLinq Plugin enabled and configured (see Configuring the DataLinq Plugin) you can apply data from a referenced source, such as a text file, a lighting control or a static mesh object to actors in your project. In the case of a static mesh object, the DataLinq component for static meshes will apply the downloaded texture only to **Texture Parameter 2D** elements of the corresponding static mesh object material, and only to the ones that have the "tg_" prefix in their name.

For instructions on enabling the DataLinq Plugin, see Enabling the Voyager Plugins.

You will need to install the XPression DataLinq Server on your network. The server is available free but requires a license to connect to it. Contact Ross Video at any of the numbers listed in the Getting Help section for assistance.

If you attempt to add a DataLinq source to your project but don't have a valid DataLinq license, you will see a message in the bottom-right corner of the screen, as shown in the image below:



DataLinq License Not Found

You will also need to add your data source to the DataLinq server in order to make it available to Voyager. See Adding a DataLinq Source to the XPression DataLinq Server.

Other topics discussed in this section include:

Adding a DataLinq Component

Adding the DataLinq Multi-Value Actor

Using a DataLinq Key

Using DashBoard DataLinq Sources.

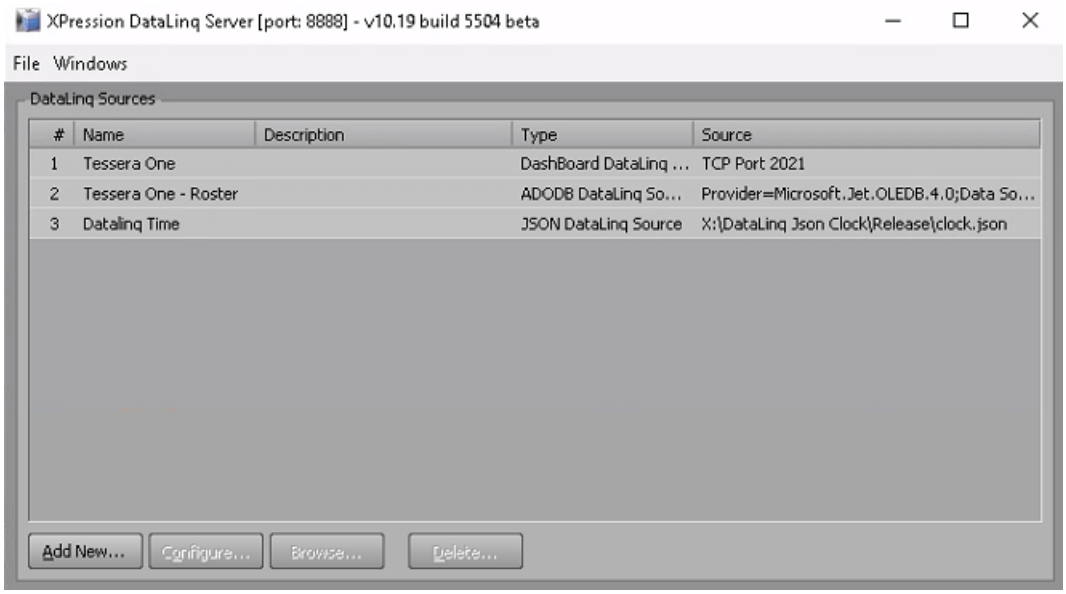
Using SQL Queries in DataLinq

Adding a DataLinq Source to the XPression DataLinq Server

Once you have your DataLinq source configured, you'll need to add it to the XPression DataLinq Server.

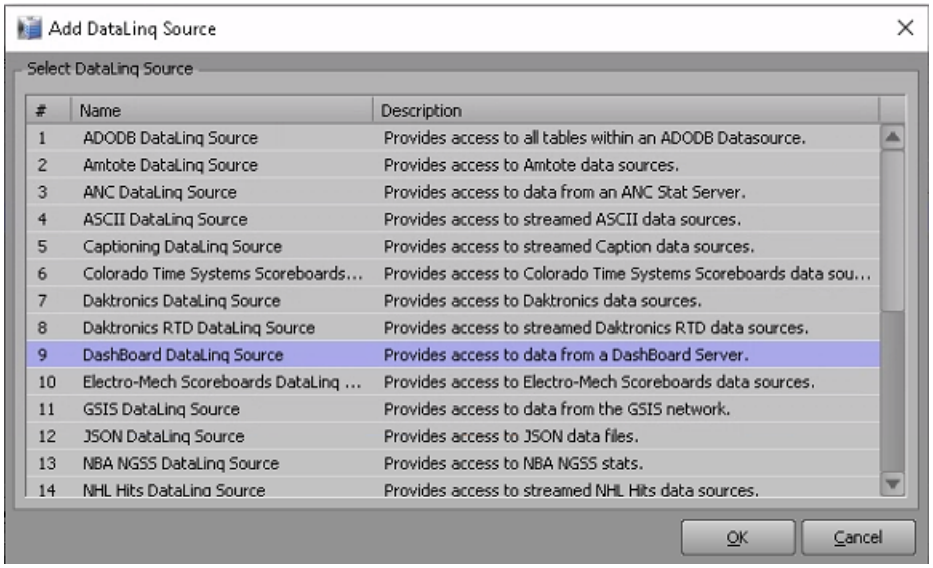
To add a DataLinq source to the XPression DataLinq Server:

1. Launch the XPression DataLinq Server.



XPression DataLinq Server

2. Select **Add New**.

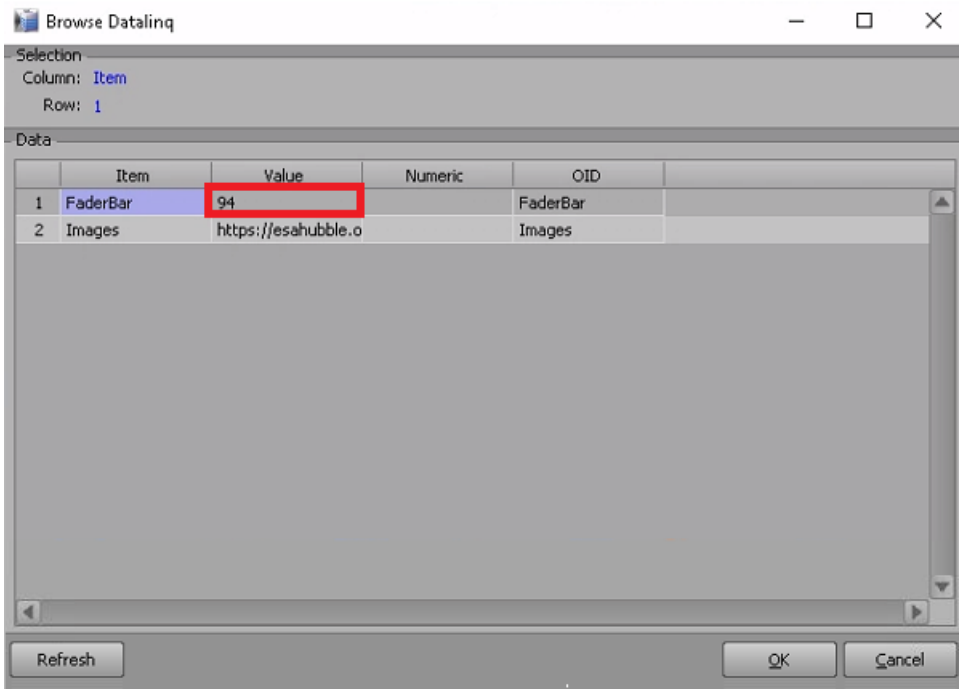


Add DataLinq Source

3. Select your DataLinq source from the list.
4. Configure your DataLinq source.

   The configuration dialog differs depending on the source selected. Once you've configured your data source, it will appear in the list.

5. Select your data source and select **Browse** to verify that the parameter value is being received.



Verify DataLinq Value

6. Select **OK** and then close the XPression DataLinq Server window.

   The XPression DataLinq Server will be minimized to the taskbar.

Adding a DataLinq Component

With the DataLinq plugin enabled and a DataLinq source added to the XPression DataLinq server, you can manually add an additional component to your project to retrieve the data from the DataLinq source.

If you only need one target in your project for the data, add a DataLinq Single-Value Component.

If you need several targets in your project for the data, add a DataLinq Multi-Value Component.

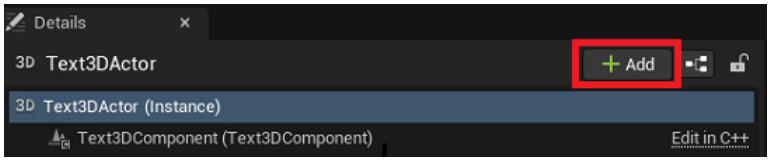
★Versions of XPression DataLinq preceding v11.1 Build 5762, require that there be 2 or more values drawn from the DataLinq source when using the **DataLinq Multi-Value Component**.

DataLinq Single-Value Component

Use this component for a single target.

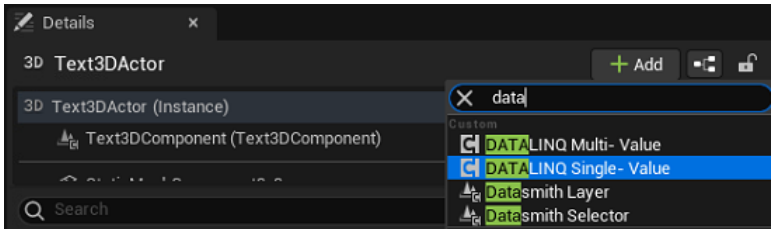
To add a DataLinq source using the DataLinq Single-Value component:

1. In the **Outliner**, select the 3D text, light or object actor to which you want to apply the data.
2. In the **Details** tab, select the **Add Component** button.



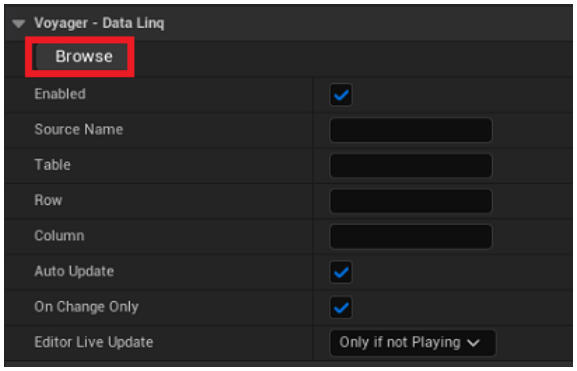
Add Component

3. In the **Search** field, start typing `DataLinq` and select **DATALINQ Single-Value**.



DataLinq - Add Single-Value Component

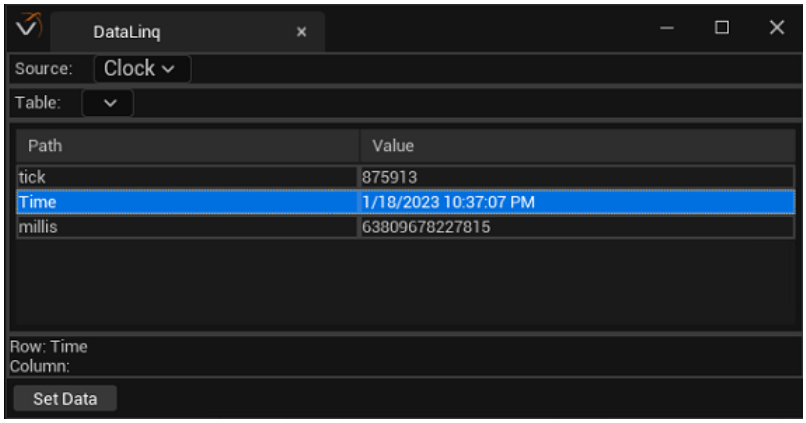
4. In the **Voyager - DataLinq** section, select the **Browse** button to open the **DataLinq** window.



DataLinq - Browse

5. From the **Source** drop-down, select the DataLinq source you want to apply to your actor.

Make sure the DataLinq source you select is running.



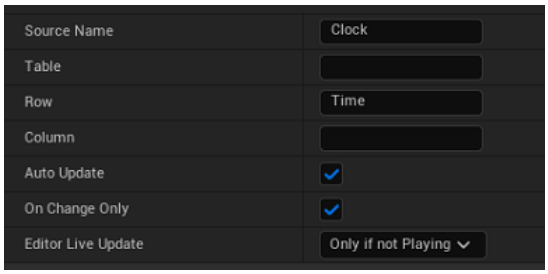
DataLinq - Select Single-Value Source

The data from the selected source is displayed in the DataLinq window.

6. If your source has more than one table, from the **Table** drop-down, select the table you want to use.
7. Then double-click a cell in the table to select the row and column you want to display in your project.

   The DataLinq window closes and the **Voyager - DataLinq** section of the **Details** tab is populated with the **Source Name** and **Table** (if applicable) from the selected source, along with the default properties.

   ★When using a DashBoard data source, the **Row** field is populated with the **OID** (Object Identifier) of the selected cell rather than the actual value.



DataLinq - Summary

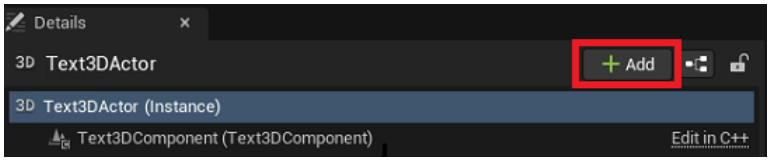
8. Set the **Update** properties as follows:
   - Select the **Auto Update** checkbox to update the data for the selected actor automatically, whether it changes or not.
   - For optimal performance, select the **On Change Only** checkbox, in addition to the **Auto Update** checkbox, to update the data for the selected actor only when it changes.
   - From the **Update in Editor** drop-down, select one of the following options to define when the DataLinq component will be updated.
     - **Never**: The object will not be updated while running in the editor.
     - **Only if not playing**: The object will be updated only if it is not in PIE mode.
     - **Always**: The object will always be updated, whether it is in editor or PIE mode.
9. Select **Save** and then **Play** to check that the data is updating correctly.

DataLinq Multi-Value Component

Use this component for multiple targets.

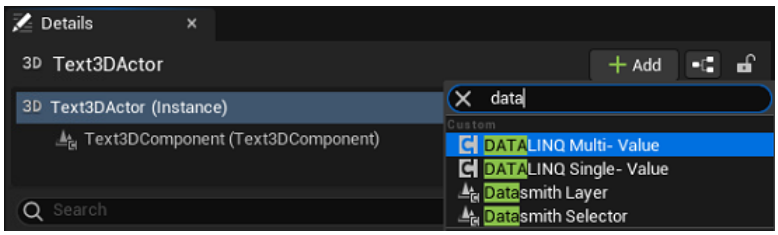
To add a DataLinq source using the DataLinq Multi-Value component:

1. In the **Outliner**, select the 3D text, light or object actor to which you want to apply the data.
2. In the **Details** tab, select the **Add Component** button.



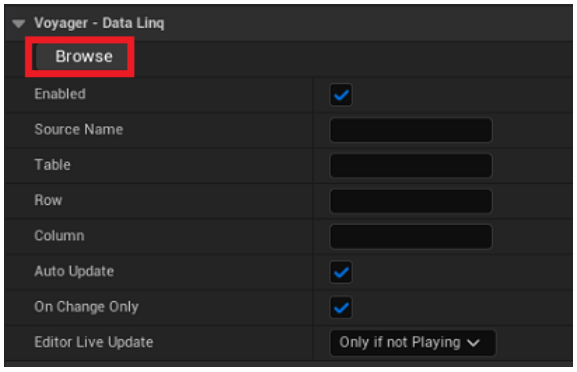
Add Component

3. In the **Search** field, start typing `DataLinq` and select **DATALINQ Multi-Value**.



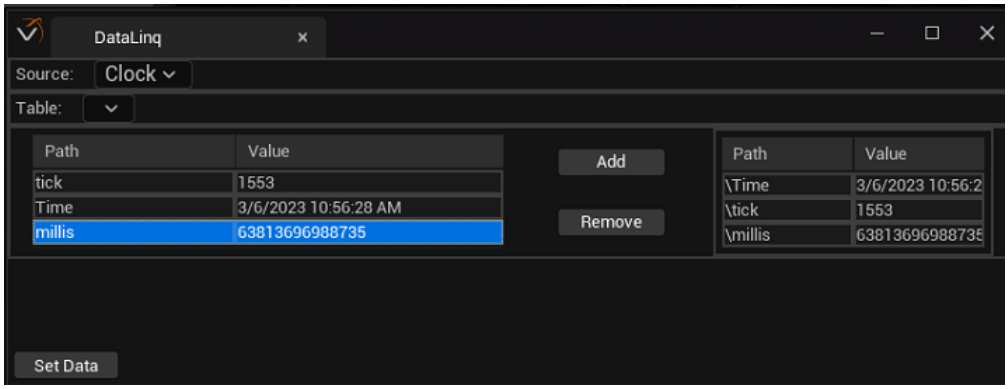
DataLinq - Add Multi-Value Component

4. In the **Voyager - DataLinq** section, select the **Browse** button to open the **DataLinq** window.



DataLinq - Browse

5. From the **Source** drop-down, select the DataLinq source you want to apply to your actor.

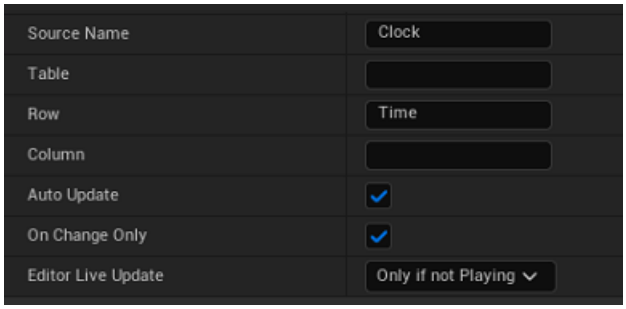


DataLinq - Select Multi-Value Source

The data from the selected source is displayed in the DataLinq window.

6. If your source has more than one table, from the **Table** drop-down, select the table you want to use.
7. Then double-click cells in the table to select the rows and columns you want to display in your project.
8. Select **Set Data** to close the window and apply the selected values.

The DataLinq window closes and the **Voyager - DataLinq** section of the **Details** tab is populated with the **Source Name** and **Table** (if applicable) from the selected source, along with the default properties.



DataLinq - Summary

9. Set the **Update** properties as follows:
   - Select the **Auto Update** checkbox to update the data for the selected actor automatically, whether it changes or not.
   - For optimal performance, select the **On Change Only** checkbox, in addition to the **Auto Update** checkbox, to update the data for the selected actor only when it changes.
   - From the **Update in Editor** drop-down, select one of the following options to define when the DataLinq component will be updated.
     - **Never**: The object will not be updated while running in the editor.
     - **Only if not playing**: The object will be updated only if it is not in PIE mode.
     - **Always**: The object will always be updated, whether it is in editor or PIE mode.

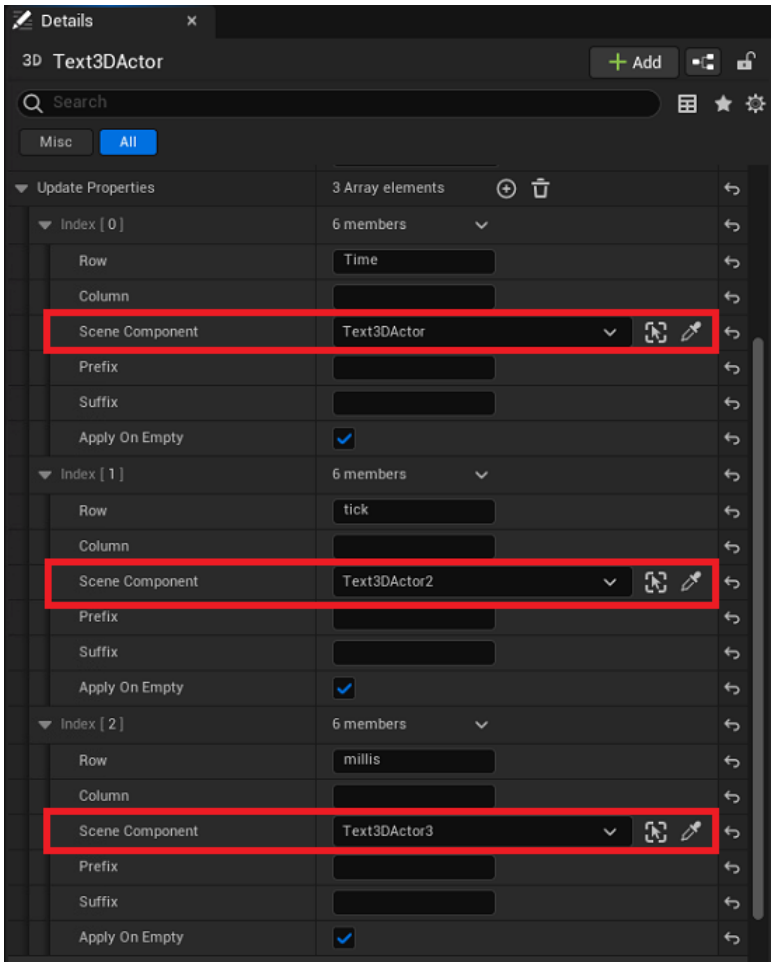
To assign values to scene components:

1. Expand the **Update Properties** section and in each **Array Element**, from the **Scene Component** drop-down, select the actor to which the DataLinq value should be applied.

   ★When using a DashBoard data source, the **Row** field is populated with the **OID** (Object Identifier) of the selected cell rather than the actual value.

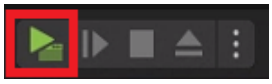
   The first value you selected from the DataLinq table will be in the **Index [0]** array element, the second will be in the **Index [2]** array element, etc.

   By default, the first value will be displayed on the 3D Text, light or object actor you selected when you added the **DataLinq Multi-Value** component.



DataLinq - Update Properties

2. Select **Save** and then **Play** to check that the data is updating correctly.



Play Button

Adding the DataLinq Multi-Value Actor

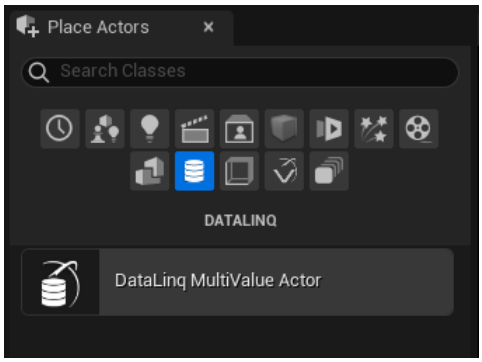
In Voyager 4.27 R3 and newer versions, a **DataLinq Multi-Value** actor has been added to the **Place Actors** tab to make adding a DataLinq source easier. You must have a DataLinq license and have enabled the DataLinq plugin, to access the actor.

You can also use the DataLinq Multi-Value actor directly in the blueprint to retrieve the source data.

See Using the DataLinq Multi-Value Actor in a Blueprint.

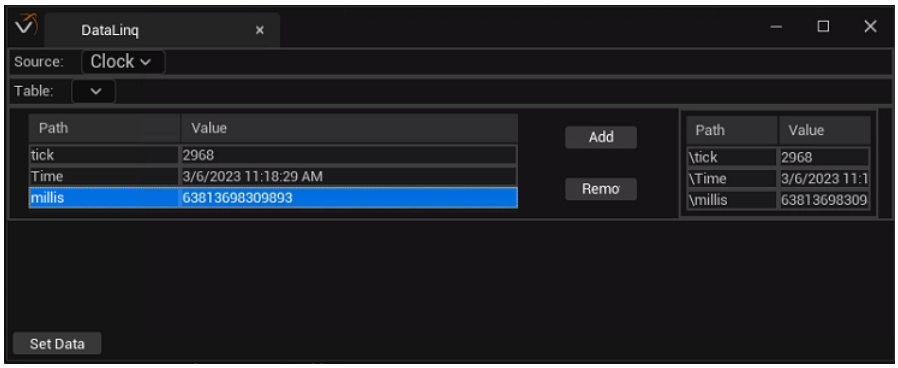
To add a DataLinq source using the DataLinq MultiValue actor:

1. In the **Place Actors** tab, from the **DataLinq** category, drag the **DataLinq MultiValue Actor** into the level.



Add DataLinq Multi-Value Actor

The **DataLinq** window opens.

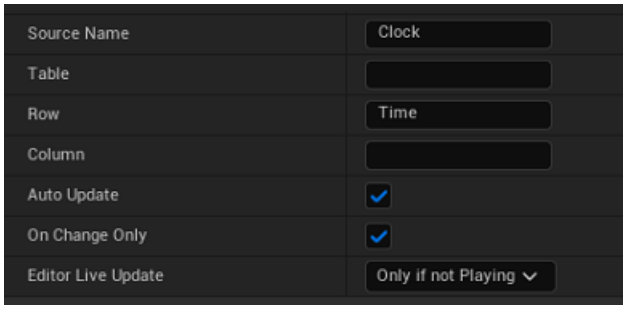


DataLinq - Select DataLinq Source

2. From the **Source** drop-down, select the DataLinq source you want to use.
3. If you have more than one table in your DataLinq source, from the **Table** drop-down, select the table you want to use.
4. Then double-click cells in the table to select the rows and columns you want to display in your project.
5. Select **Set Data**.

6. In the **Details** tab, select the **DataLinqMultiValueComponent**.

   The DataLinq window closes and the **Voyager - DataLinq** section of the **Details** tab is populated with the **Source Name** and **Table** (if applicable) from the selected source, along with the default properties.



   DataLinq - Summary

7. Expand the **Voyager - DataLinq** section and set the **Update** properties as follows:

   - Select the **Auto Update** checkbox to update the data automatically, whether it changes or not.
   - For optimal performance, select the **On Change Only** checkbox, in addition to the **Auto Update** checkbox, to update the data for the selected actor only when it changes.
   - From the **Update in Editor** drop-down, select the update behavior.
     - **Never**: The object will not be updated while running in the editor.
     - **Only if not playing**: The object will be updated only if it is not in PIE mode.
     - **Always**: The object will always be updated, whether it is in editor or PIE mode.

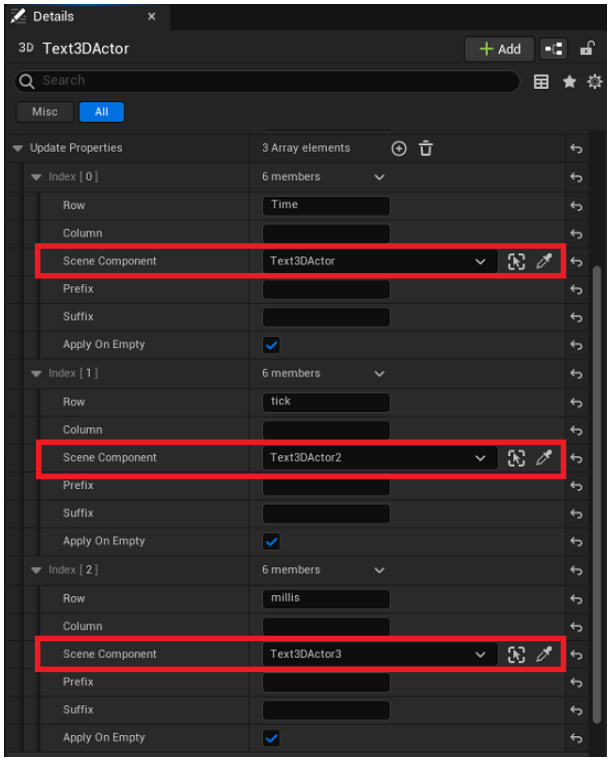
8. Expand the **Update Properties** section and in each **Array Element**, from the **Scene Component** drop-down, select the actor to which the DataLinq value should be applied.

   If you are using a blueprint to update the properties from your data source, you can skip this step. See Using the DataLinq Multi-Value Actor in a Blueprint for further information.

   ★When using a DashBoard data source, the **Row** field is populated with the **OID** (Object Identifier) of the selected cell rather than the actual value.

   The first value you selected from the DataLinq table will be in the **0 Array Element**, the second will be in the **1 Array Element**, etc.

   By default, the first value will be displayed on the 3D Text, light or object actor you selected when you added the DataLinq Multi-Value component.



DataLinq - Update Properties

9. Select **Save** and then **Play** to check that the data is updating correctly.

Play Button

Using a DataLinq Key

Starting with Voyager 7.0, you can use the **DataLinq Key** feature to replace data in your DataLinq source. The **DataLinq Key** can then be used in a blueprint to update that data from Lucid Studio, Voyager Trackless, DashBoard or other applications.

To use this feature, you will need to add your DataLinq source to the XPression DataLinq Server (see instructions here). Then follow the instructions in the next sections to add and configure a **DataLinq Multi-Value** actor, use a **DataLinq Key** in the **DataLinqMultiValueComponent** and create event graphs in the level blueprint to update the data.

A **DataLinq Key** can be used in an event graph to update the data in a column or row of your DataLinq source or it can be used to update an entire table. The **DataLinq Key** can be any text you want to use to describe the data it is replacing, e.g., State, State_Code, Data_Set. It needs to use the syntax **<% DataLinq Key%>** or <**Attribute=%DataLinq_Key%>**.

★It is recommended that you use XPression DataLinq Server v11.5 build 5816 or later.

The following steps take you through this process for using the DataLinq Key in a blueprint to update the value of a column, row or table:

Add and configure a DataLinq Multi-Value Actor

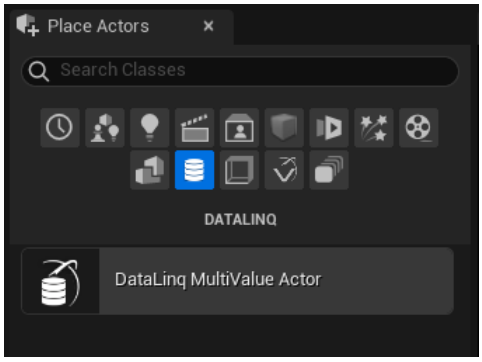
Enter a DataLinq Key in the DataLinqMultiValueComponent

Create an Event Graph to Update a DataLinq Key Value

Update Data in a Text3D Actor

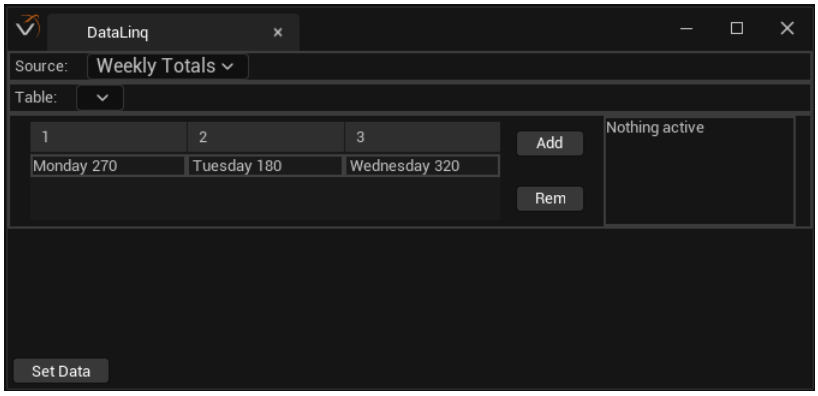
To add and configure a DataLinq Multi-Value Actor:

1. In the **Place Actors** tab, from the **DataLinq** category, drag the **DataLinq MultiValue Actor** into the level.



Add DataLinq Multi-Value Actor

The **DataLinq** window opens with the most recently selected source displayed.



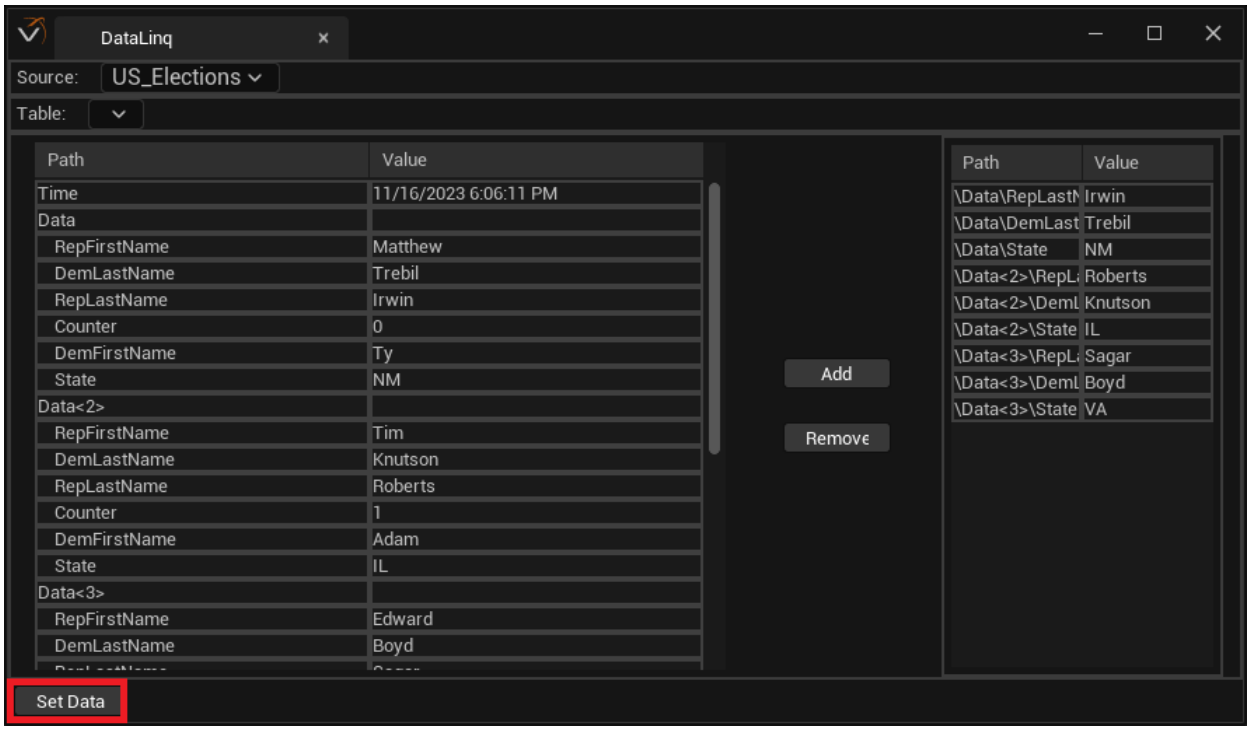
DataLinq - Select DataLinq Source

2. From the **Source** drop-down, select the DataLinq source you want to use.
3. Then select the cell(s) in the table containing the data you want to display and select **Add** to add the data to the list on the right.

   Alternatively, you can double-click a cell to add the data to the list.
4. Continue adding data from the table to the list until you have everything you want and then select **Set Data**.

   In the image below, the selected data is the Republican last name, Democrat last name and the state.

   Note that the first set of data does not indicate **<1>**. The number 1 is assumed.



DataLinq Key - Select Data

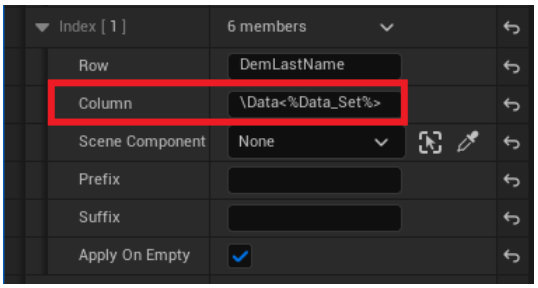
The **DataLinq** tab closes.

5. In the **Outliner**, right-click the **DataLinq Multi-Value Actor** you just created, select **Edit > Rename** and give it a meaningful name.

To enter a DataLinq Key in the DataLinqMultiValueComponent:

1. In the **Outliner**, select the **DataLinq Multi-Value Actor** you just created.
2. In the **Details** tab, select the **DataLinqMultiValueComponent**.
3. In the **Voyager - DataLinq** section of the **Details** tab, do the following:
   - If you're using the **DataLinq Key** to replace row or column data, expand **Update Properties** and expand each of the **Indexes**.
   - In the **Column** (or row) field of each **Index**, replace the number of the array or data set with a **DataLinq Key**, using either the syntax **<%DataLinq Key%>** or <**Attribute=%DataLinq_Key%>**.

     For the **Indexes** for the first set of data, the number 1 is assumed, so you just add the **DataLinq Key**.

     DataLinq Key - Enter DataLinq Key

     In the image above, the data set is **\Data**, and the **DataLinq Key** is **%Data_Set%**.

     This will return the data contained in that data set.

     Alternatively, you can use **\Data<Attribute=%DataLinq_Key%>** where the **Attribute** is a column or row **Name** and **%DataLinq_Key%** is the text you use to replace the data in that column or row.

     OR

   - If you're using the **DataLinq Key** to replace the name of a table, scroll down to the **Table** field (after all the **Indexes**).
   - In the **Table** field, replace the table name with the **DataLinq Key** you want to use to represent the table (in this example, **%STATE_CODE%**).

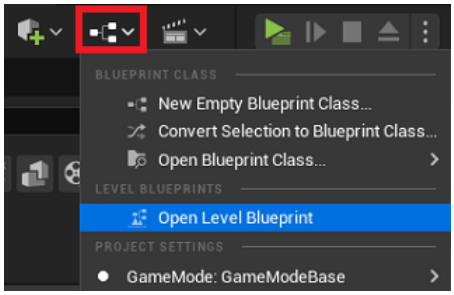
     This will return all the data in the table for each State code.

     DataLinq Key - Enter DataLinq Key (Table)

To create an event graph to update a DataLinq Key value:

1. Select the **Blueprint** icon and select **Open Level Blueprint**.

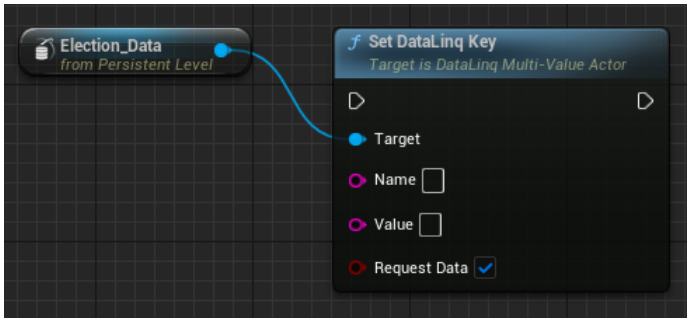


Open Level Blueprint

2. From the **Outliner**, drag the **DataLinqMultiValueActor** you created in Using the DataLinq Key into the blueprint.

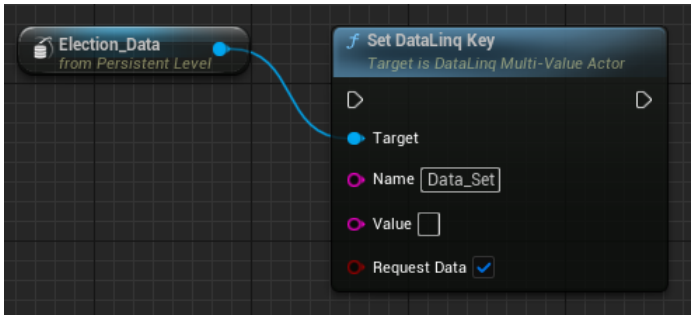
   For this example, the **DataLinqMultiValueActor** is called **Election_Data**.

3. Using the **Left Mouse Button**, drag off from the **Output** pin of the **DataLinqMultiValueActor** and in the **Search** field, begin typing `set datalinq key` and then select the **Set DataLinq Key** node.



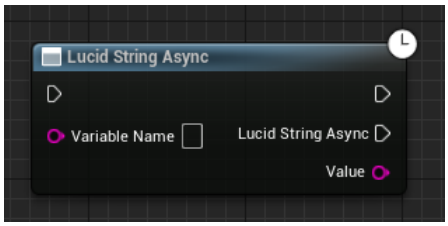
DataLinq Blueprint - Add Set DataLinq Key Node

4. In the **Name** field of the **Set DataLinq Key** node, enter the **DataLinq Key** (without the percent symbols) you created in the previous procedure.



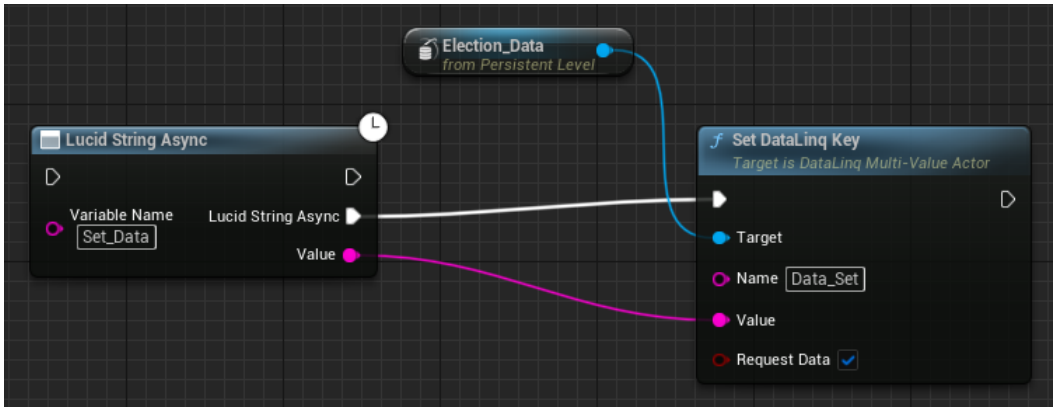
DataLinq Blueprint - Enter DataLinq Key

5. Right-click in an empty area of the blueprint and add a blueprint node to execute the **Set DataLinq Key** method (e.g., a Keyboard Event node, RossTalk command, Lucid String Async, etc.).



DataLinq Blueprint - Add Lucid String Async Node

6. If using a **Lucid String Async** node, in the **Variable Name** field, enter the name you will use in Lucid Studio to identify the node (e.g., **Set_Data**).
7. Connect the **Lucid String Async Output** pin of the **Lucid String Async** node to the **Exec Input** pin of the **Set DataLinq Key** node.
8. Then connect the **Value** pin of the **Lucid String Async** node to the **Value** pin of the **Set DataLinq Key** node.



DataLinq Blueprint - Connect Lucid String Async Node

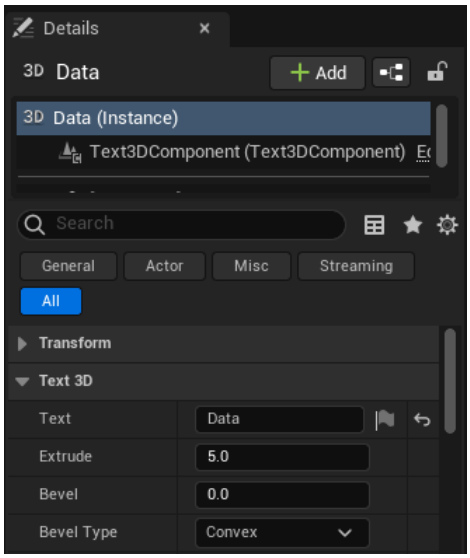
Updating Data in a Text3D Actor

Once you have an event graph to update the column or row data from your data source, you can output the updated data (to a **Text3D** actors in this example).

First you'll need to add the **Text3D** actor to the project and then create an event graph to update the data in the **Text3D** actor.

To add a Text3D actor to your project:

1. In the **Place Actors** tab to the left of the **Viewport**, in the **Search** field, start typing `text3d` and select and drag a **Text3D** actor into the **Viewport**.
2. In the **Outliner**, right-click the **Text3D** actor and select **Edit > Rename** to give the actor a meaningful name (in this example, **Data**).
3. With the **Text3D** actor selected, in the **Details** tab, select **All** and in the **Text 3D** section, in the **Text** field, enter the name of the data to be displayed in this actor (typically the same name as the **Text3D** actor itself).

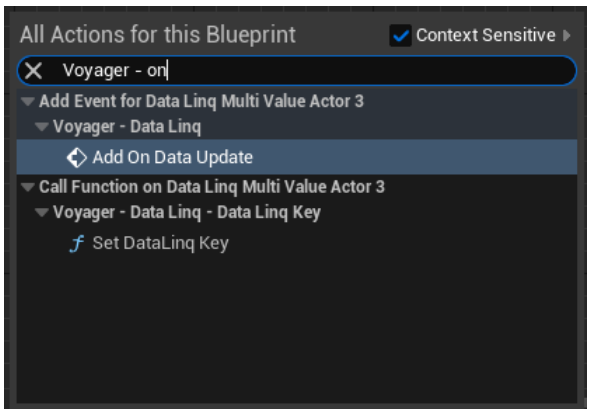


DataLinq Blueprint - Configure Text3D Actor

4. Use the other options in the **Text 3D** section to modify the appearance of the text.
5. Position the **Text3D** actor where you want it in the **Viewport**.

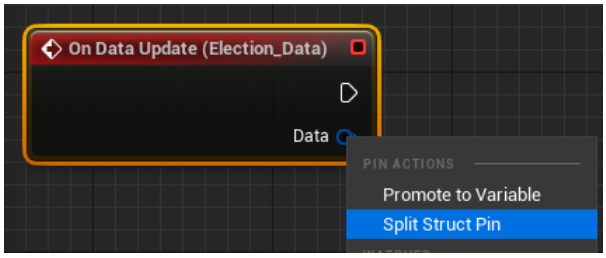
To create an event graph to update data in the Text3D actor:

1. With the **DataLinqMultiValueActor** you created selected in the **Outliner**, right-click in an empty area of the blueprint and in the **Search** field, begin typing `Voyager – on Data Update` and then select the **Add On Data Update** node.



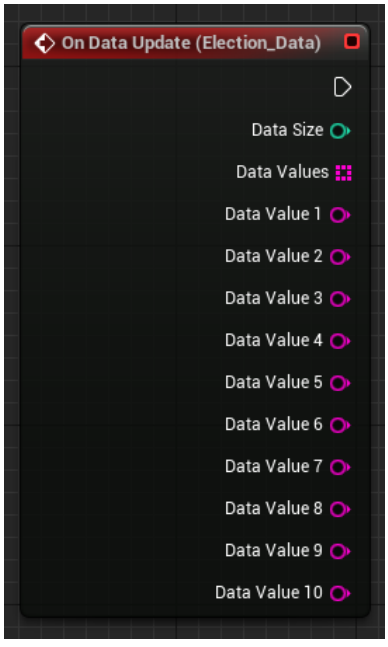
DataLinq Blueprint - Add Add On Data Update Node

2. Right-click the **Data** pin of the **Add On Data Update** node and select **Split Struct Pin**.



DataLinq Blueprint - Split Struct Pin

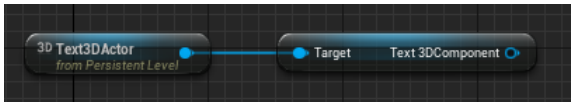
This expands the node with multiple **Data** pins.



DataLinq Blueprint - Expanded On Data Update Node

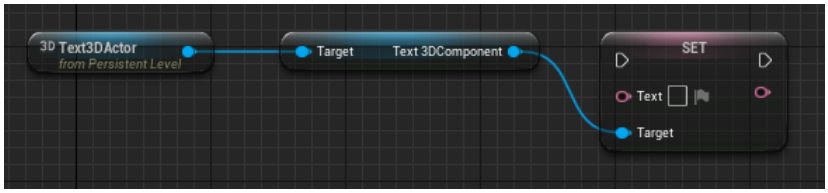
3. In the **Outliner**, select and drag the **Text3D** actor you created into the blueprint.

4. Left-click and drag off the **Output** pin of the **Text3D** actor node, type `Get Text 3D`, scroll down to the bottom of the menu and select the **Get Text 3D Component** node.



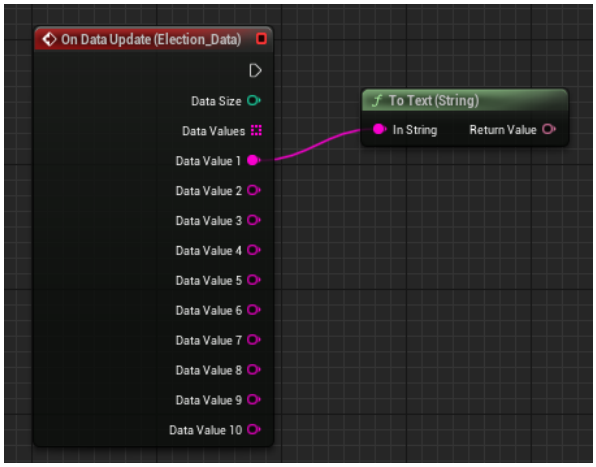
DataLinq Blueprint - Add Get Text 3D Component Node

5. Left-click and drag off the **Text 3D Component** node, type `Set Text`, and select the **Set Text** node.



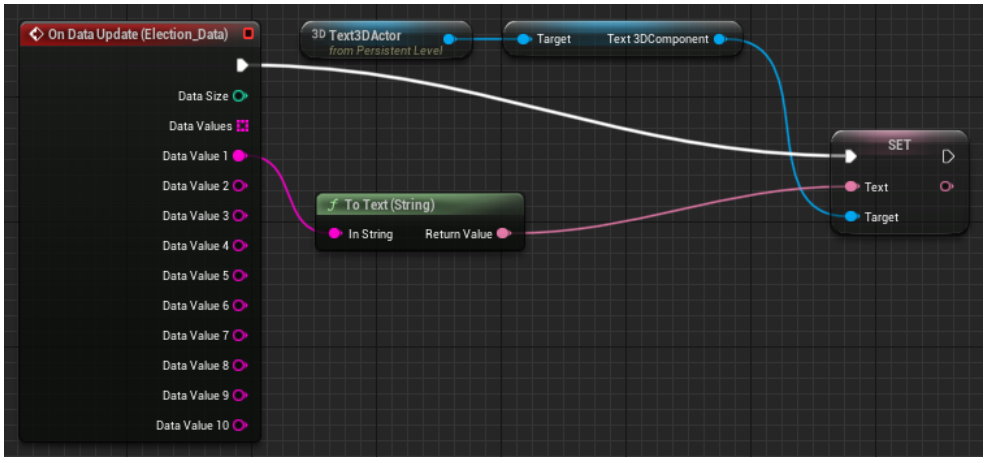
DataLinq Blueprint - Add Set Text Node

6. Left-click and drag off the first **Data Value** pin of the **On Data Update** node and in the **Search** field, begin typing `To Text` and select the **To Text (String)** node.



DataLinq Blueprint - Add To Text (String) Node

7. Connect the **Return Value** pin of the **To Text (String)** node to the **Text** pin of the **Set Text** node.
8. Connect the **Exec Output** pin of the **On Data Update** node to the **Exec Input** pin of the **Set Text** node.



DataLinq Blueprint - Update Text3D Actor Event Graph

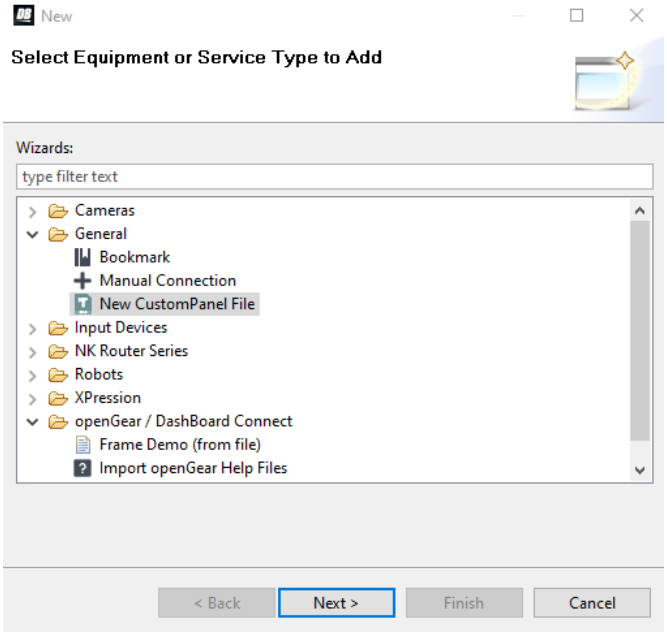
9. Save and compile your blueprint.

Using DashBoard DataLinq Sources

To use DataLinq sources from DashBoard, you'll need to create a custom DashBoard panel for Voyager. In this panel, you'll add the data source to be applied to the Voyager actor. This could be a number control, an image, or text. Once you have the source configured, you'll add your DashBoard DataLinq source to the XPression DataLinq Server.

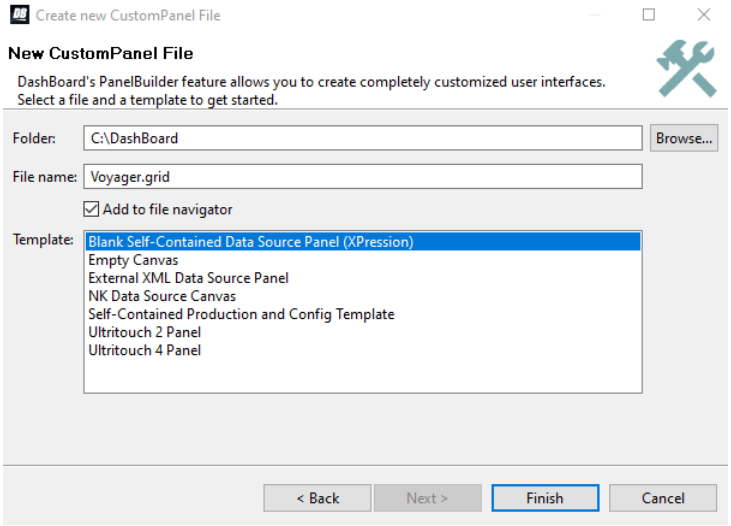
To create a custom DashBoard panel:

1. Launch DashBoard.
2. Select the **+** icon at the top of the **Basic Tree View**.
3. In the **Select Equipment or Service Type to Add** window, select **General > New CustomPanel File** and select **Next**.



DashBoard DataLinq - Select New CustomPanel File

4. Keep the **Folder** as is or select the **Browse** button to navigate to a new folder.



DashBoard DataLinq - Create CustomPanel File

5. In the **File Name** field, enter a name for your custom panel (e.g., Voyager.grid).
6. From the **Template** list, select **Blank Self-Contained Data Source Panel (XPression)** and select **Finish**.

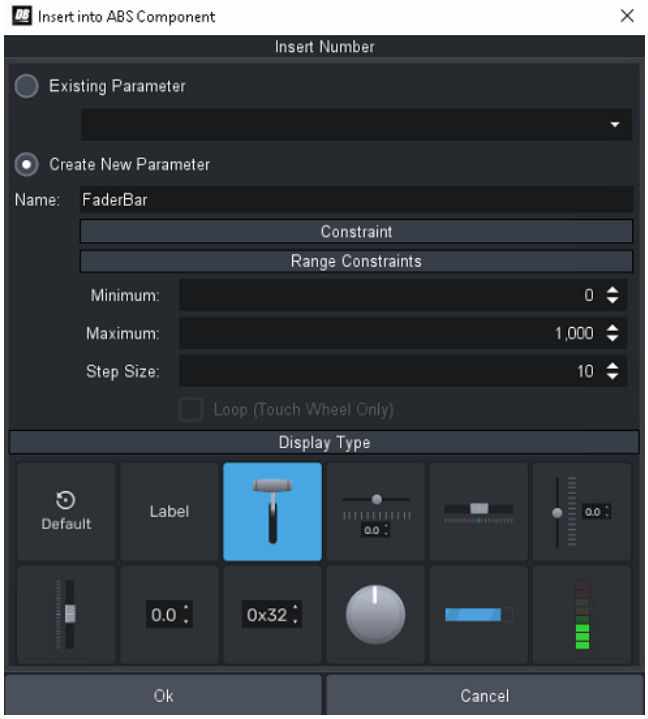
To add a light control component to the panel:

1. Select **PanelBuilder Edit Mode** and double-click the panel.
2. In the **Abs Attributes** tab, set the **XPression DashBoard Linq Port** and select the **Enable Streaming** checkbox.
3. Select **Apply and Close**.
4. Select **Params > Number** and drag a rectangle in the panel to add a number parameter.

DashBoard DataLinq - Add a Number Parameter

The **Insert into ABS Component** window opens.

5. In the **Insert into ABS Component** window, select the **Create New Parameter** radio button.



DashBoard DataLinq - Configure Number Parameter

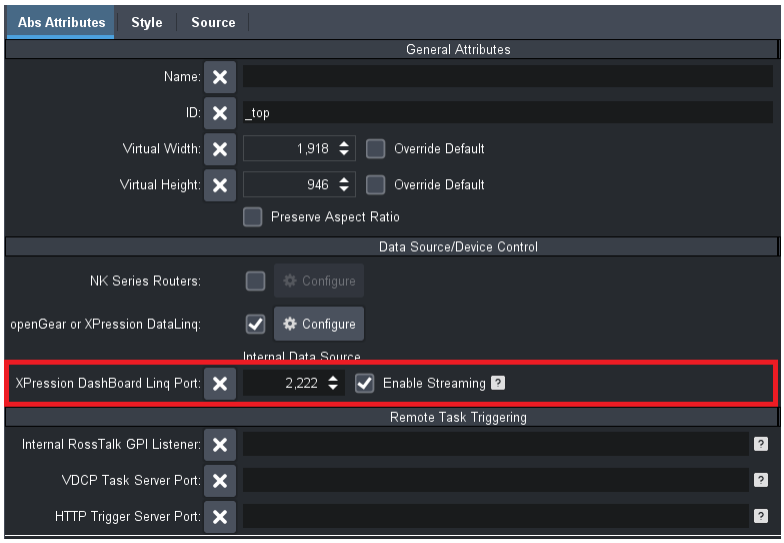
6. Enter a **Name** (e.g., FaderBar) for the parameter.
7. Set the **Minimum** and **Maximum** values and the **Step Size**.
8. Select the **Display Type** (e.g., Fader Bar).
9. Then select **Ok** and exit from **PanelBuilder Edit Mode**.

Continue with Adding a DataLinq Source to the XPression DataLinq Server.

To add an image component to the panel:

1. Select **PanelBuilder Edit Mode** and double-click the panel.

   The **Edit Component** window opens.



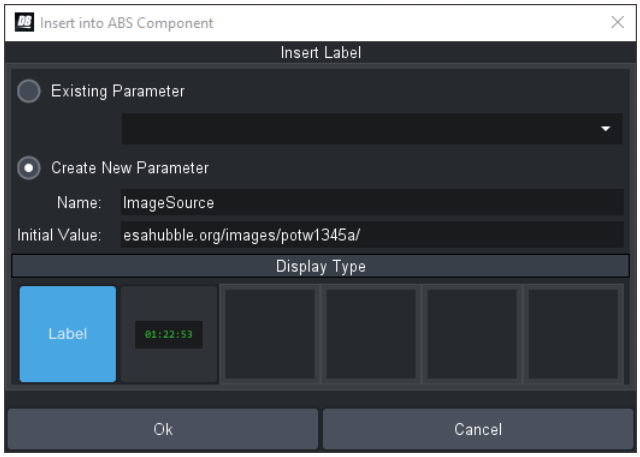
DashBoard DataLinq - Edit Component

2. In the **Abs Attributes** tab, set the **XPression DashBoard Linq Port** and select the **Enable Streaming** checkbox.
3. Select **Apply** and **Close**.
4. Select **Params > Label** and drag a rectangle in the panel to define the label area.

DashBoard DataLinq - Add an Image Source Parameter

The **Insert into Abs Component** window opens.

5. In the **Insert into ABS Component** window, select the **Create New Parameter** radio button.



DashBoard DataLinq - Configure Image Source Parameter

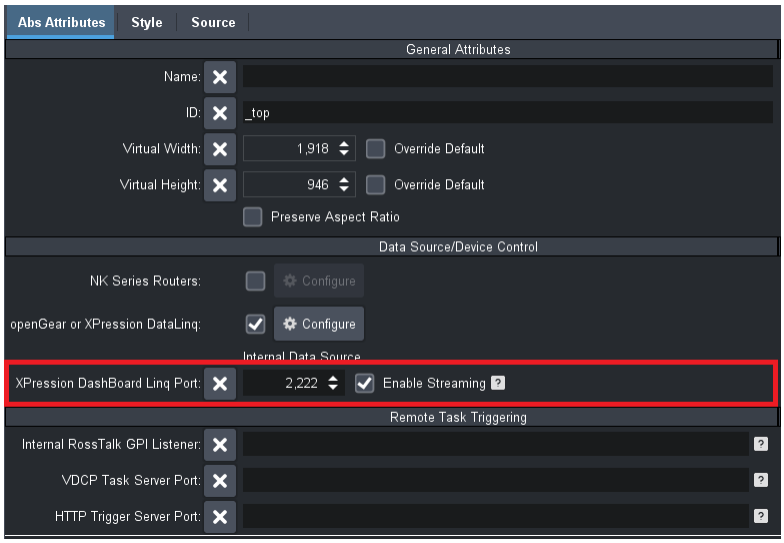
6. Enter a **Name** (e.g., ImageSource) for the parameter.
7. In the **Initial Value** field, enter a **URL** to the image you want to apply to your static mesh object.
8. In the **Display Type** section, select **Label.**
9. Then select **Ok** and exit from **PanelBuilder Edit Mode**.

Continue with Adding a DataLinq Source to the XPression DataLinq Server.

To add a text component to the panel:

1. Select **PanelBuilder Edit Mode** and double-click the panel.

The **Edit Component** window opens.



DashBoard DataLinq - Edit Component

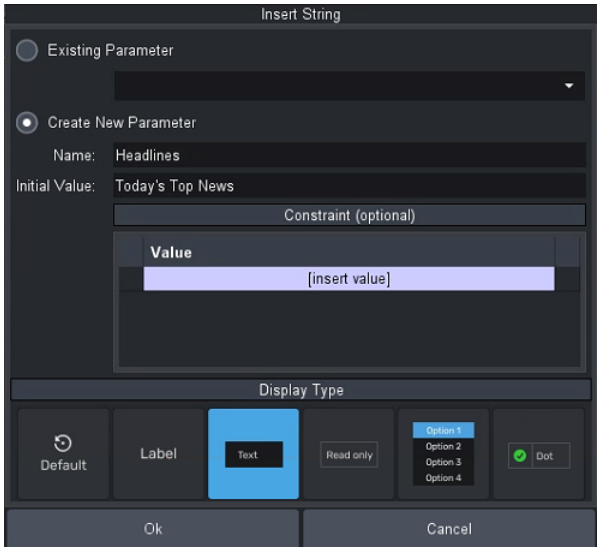
2. In the **Abs Attributes** tab, set the **XPression DashBoard Linq Port** and select the **Enable Streaming** checkbox.
3. Select **Apply and Close**.

4. Select **Params > Text** and drag a rectangle in the panel to define the text area.

DashBoard DataLinq - Add a Text Parameter

The **Insert into ABS Component** window opens.

5. In the **Insert into ABS Component** window, select the **Create a New Parameter** radio button.



DashBoard DataLinq - Configure Text Parameter Insert into ABS Component

6. Enter a name for the parameter.
7. In the **Initial Value** field, enter the text that you want to appear first.
8. In the **Display Type** section, select the format in which you want your text to appear.
9. Then select **Ok** and exit from **PanelBuilder Edit Mode**.

   The text you entered in the **Initial Value** field appears in the text box and can now be edited.

Using SQL Queries in DataLinq

With DataLinq enabled and an XPression ADODB DataLinq source configured, you can use Structured Query Language (SQL) to retrieve data from a table in the DataLinq source.

To select your DataLinq Source:

1. In Voyager, from the **Place Actors** tab, select and drag the **DataLinq Multi-Value Actor** into the level.
2. In the **DataLinq Multi-Value Actor**, from the **Source** drop-down, select your ADODB DataLinq source.
3. From the **Table** drop-down, select the table from the ADODB DataLinq source.
4. Then select **Set Data** to save your selections and close the **DataLinq Multi-Value Actor**.

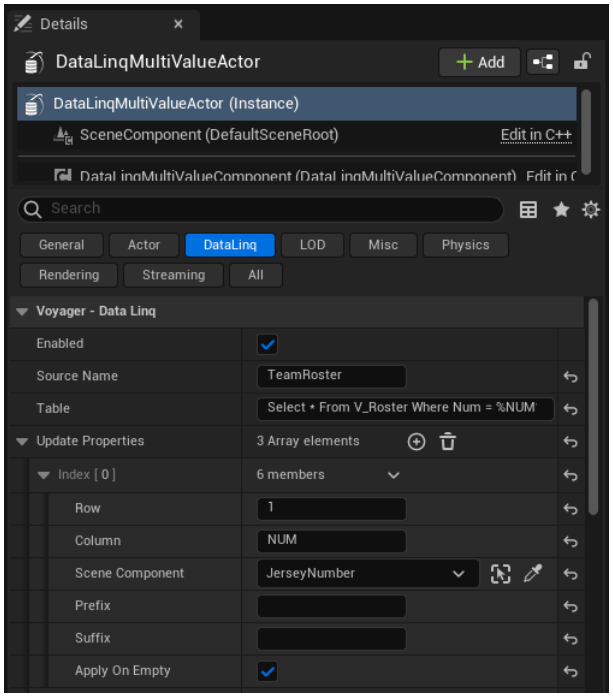
To add 3D Text Actors:

1. From the **Place Actors** tab, select and drag one **Text 3DActor** into the level for each piece of text data you want to display.
2. In the **Outliner**, rename the **Text 3DActors** appropriately (e.g., JerseyNumber, FirstName, LastName).

To configure the DataLinq Multi-Value Actor:

1. In the **Outliner**, select the **DataLinq Multi-Value Actor**.
2. In the **Details** tab, select the **DataLinqMultiValueActor (Instance)**.
3. In the **Voyager - DataLinq** section, in the **Source Name** field, enter the name of your DataLinq source.
4. In the **Table** field, enter the SQL query that will identify the column, (in this example, `Select * from V_Roster Where Num - %NUM%`).

   In this example, **V_Roster** is the name of the table in the DataLinq source and **Num** is the name of the column.



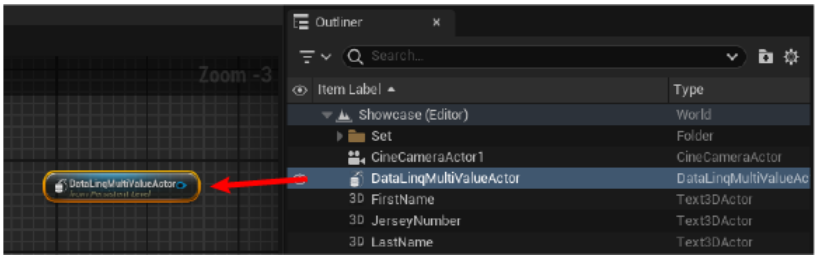
DataLinqMultiValueActor - Settings

5. In the **Update Properties** line, select the **+** icon to add as many array elements as you need, one for each piece of data (for example, for three players, each with three pieces of data, you would add 9 array elements).
6. Expand **Index 0** and do the following:
   - In the **Row** field, enter **1**.
   - In the **Column** field, enter the name of the column (in this example **NUM**).
   - From the **Scene Component** drop-down, select the **Text 3DActor** that will display the data in the selected column (in this example, **JerseyNumber**).
7. Expand each remaining **Index** and repeat the process, entering the **Row** number, the **Column** name and selecting the **Scene Component** for each.

   For this example, there will be three indexes for **Row 1**, to show the player's jersey number, and first and last name. For the next player, you would add three more indexes, each with **2** in the **Row** field.

To build the blueprint:

1. Select the arrow beside the **Blueprint** icon and select **Open Level Blueprint**.
2. Minimize the blueprint, so that you can see the **Outliner** in the main UI.
3. From the **Outliner**, drag the **DataLinqMultiValueActor** into the blueprint to add a reference node.



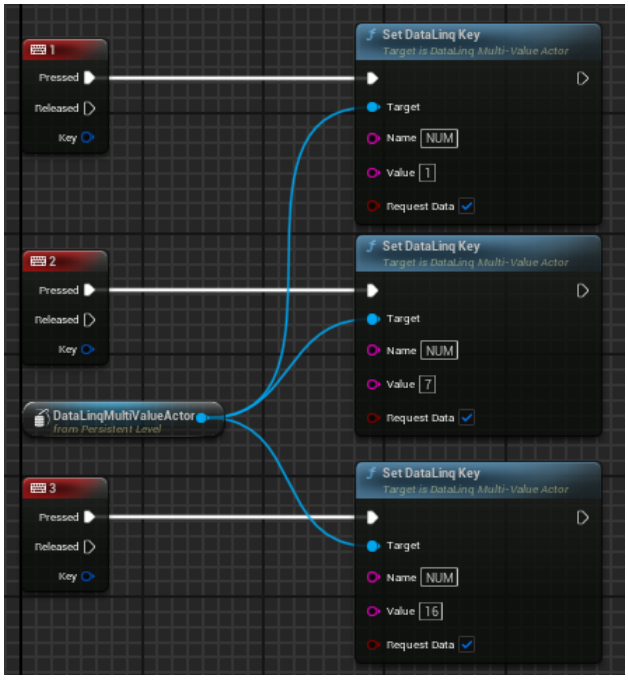
Drag DataLinqMultiValueActor into Blueprint

4. Drag out from the **DataLinqMultiValueActor** pin and in the **Search** field, begin typing `Set DataLinq Key`.
5. In the **Set DataLinq Key** node, do the following:
   - In the **Name** field, enter **NUM**.
   - In the **Value** field, enter the number of the first piece of data, in this example, the jersey number).
6. Drag out from the **DataLinqMultiValueActor** pin again to add more **Set DataLinq Key** nodes and enter the information in the **Name** and **Value** fields.

   The **Name** field will always be **NUM** and the **Value** field will be the number of the first piece of data.
7. Right-click in a blank area of the blueprint and in the **Search** field, enter `1` to add a **Key Press Event** node.

8. Add additional **Key Press Event** nodes for each **Set DataLinq Key** node that you've added.
9. Connect the **Pressed** pin of each **Key Press Event** node to the **Input** pin of the corresponding **Set DataLinq Key** node.

   The blueprint for this example (for one player) would look like this:



DataLinq SQL Query Blueprint

Using NDI®Media I/O

NDI inputs and output are configured in the **Media Profile**. If you require additional outputs, you can use NDI Broadcast Actors.

★The number of inputs and outputs available depends on your Voyager NDI Streams License.

★NDI Receive Actor and NDI Broadcast Actor rendering are not the same as the **Media I/Os** configured in the **Media Profile**. The **NDI Broadcast Actor** is useful for getting preview outputs from different perspectives, but should not be used in place of the **Media Output**.

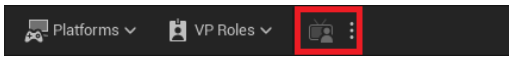
Before getting started, from the official NDI website, download either of the following:

- NDI Tools version 6.0.1
- NDI SDK 6.0.1

For information on how to set up and use **NDI Tools**, see the **NDI Tools** documentation on the official NDI website.

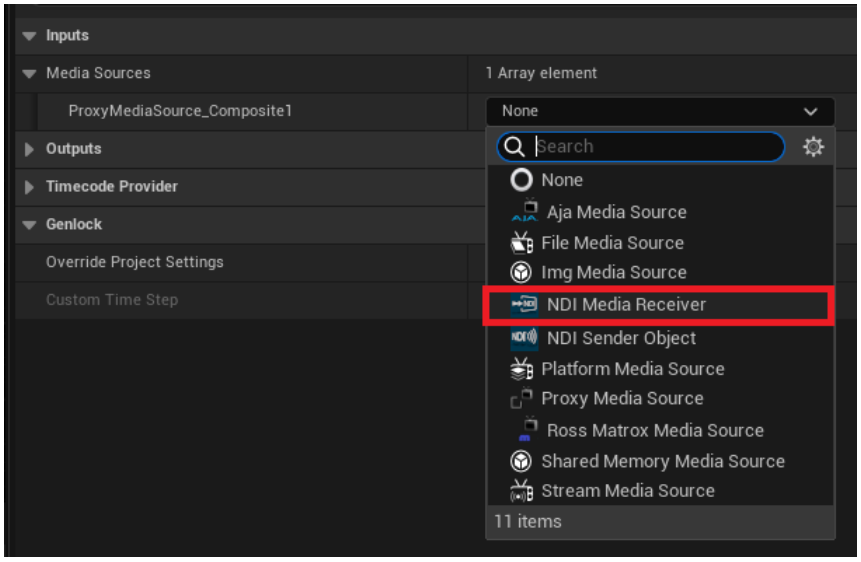
To configure an NDI input in the Media Profile:

1. In Voyager, in the main toolbar, select the **Media Profile** icon.



Media Profile Icon

2. In the **Media Profile** editor, expand **Media Sources** for the input source.
3. Expand **ProxyMediaSource** and from the drop-down, select **NDI Media Receiver**.



Select NDI Media Receiver

4. Expand the **Settings** section and from the **Connection** drop-down, select the NDI source you want.

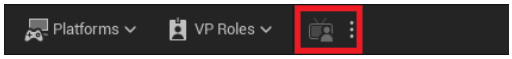
   All NDI sources in the network are automatically detected and displayed in the **Connection** drop-down.

5. Select **Save** and close the **Media Profile** editor.

   The changes have been saved and the project now displays the NDI source in the composite plane.

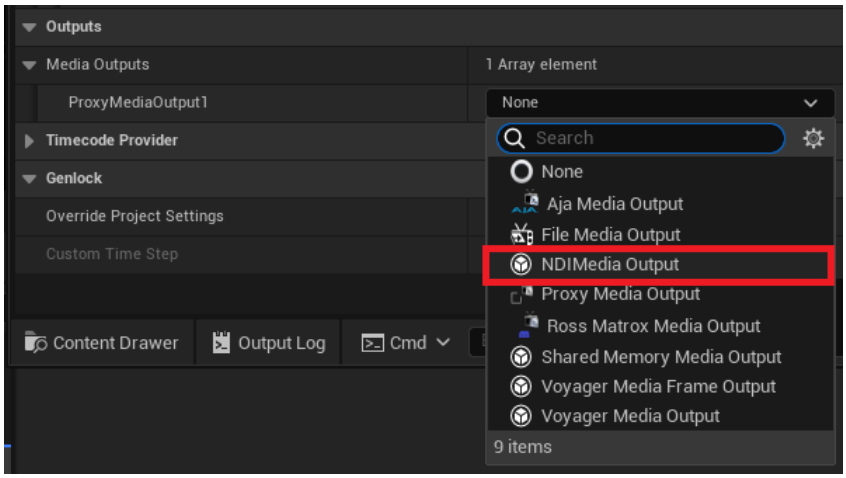
To configure the NDI output in the Media Profile:

1. In Voyager, in the main toolbar, select the **Media Profile** icon.



Media Profile Icon

2. In the **Media Profile** editor, expand **Outputs** and **Media Outputs**.
3. From the **ProxyMediaOutput1** drop-down, select **NDIMedia Output**.



Select NDI Media Output

4. Expand **Broadcast Settings** and configure the output settings as follows:

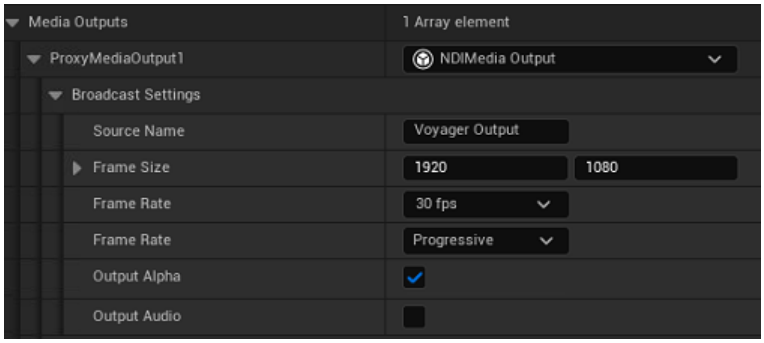
   Source Name - enter a name for the source (e.g., Voyager Output)

   Frame Size - enter the pixel values of the frame (e.g., x=1920 and y=1080)

   Frame Rate - from the drop-down, select the frame rate.

   Output Alpha - select to enable viewing the **Alpha** channel.

   Output Audio - select if your video includes audio and you want it included in the output.



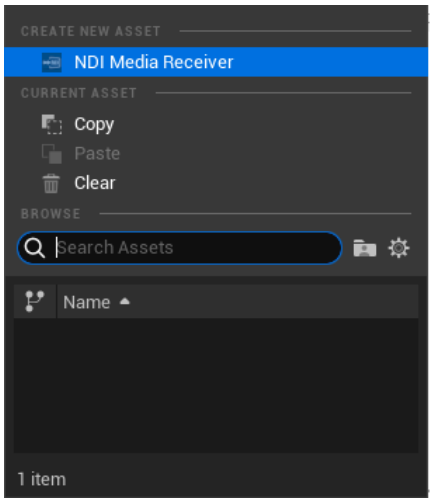
Broadcast Settings

★The Voyager Operator **Compositing Mode** needs to be set to **External** to view the **Alpha** channel.

5. Select **Save** and close the **Media Profile** editor.

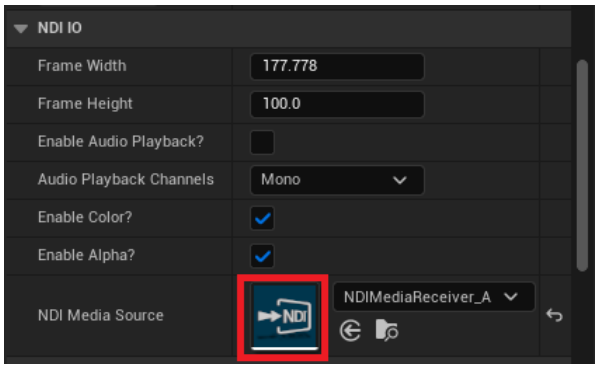
To configure an NDI Receive Actor:

1. From the **Place Actors** panel, select the **NDI** category.
2. Select the **NDI Receive Actor** to add an input and drag it into the Viewport.
3. Position the actor in the Viewport.
4. In the **Outliner**, select the **NDIReceiveActor** and rename it to something meaningful.
5. With the **NDIReceiverActor** selected in the **Outliner**, in the **Details** panel, scroll down to the **NDI IO** section, and from the **NDI Media Source** drop-down, select **Create New Asset > NDI Media Receiver**.



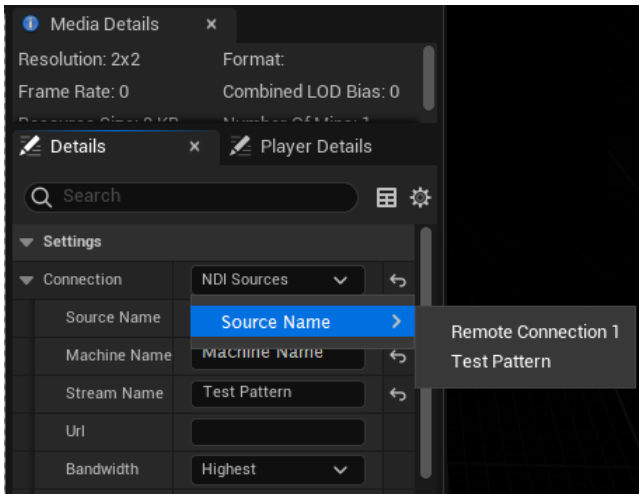
Create New NDI Media Receiver

6. In the **Save Asset As** window, navigate to the folder where you want to save the asset and in the **Name** field, enter a name for the asset and select **Save**.
7. With the **NDIReceiveActor** selected in the **Outliner**, in the **Details** tab, in the **NDI IO** section, double-click the **NDI Media Receiver** icon to open the editor.



Open NDI Media Receiver Editor

8. In the editor, in the **Details** panel, expand **Connection** and from the **NDI Sources** drop-down, select your media source.



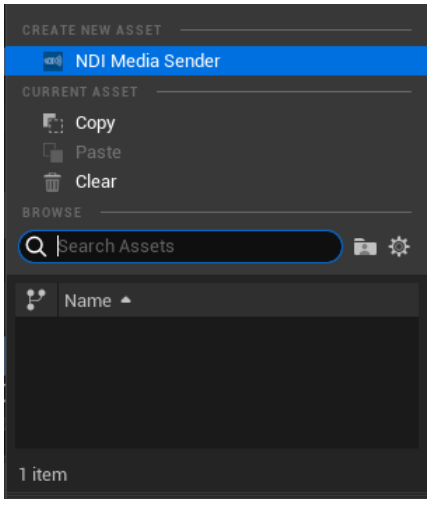
NDI Media Receiver Editor - Select NDI Source

The **Source Name**, **Machine Name**, and **Stream Name** are automatically populated.

9. Select **Save** and close the editor.

To configure an NDI Broadcast Actor:

1. From the **Place Actors** panel, select the **NDI** category.
2. Select the **NDI Broadcast Actor** to add an output and drag it into the Viewport.
3. Position the **NDI Broadcast Actor** so that it is projecting onto the **NDI Receive Actor**.
4. In the **Outliner**, select the **NDIBroadcastActor** and rename it to something meaningful.
5. With the **NDIBroadcastActor** selected in the **Outliner**, in the **Details** panel, scroll down to the **NDI IO** section, and from the **NDI Media Sender** drop-down, select **Create New Asset > NDI Media Sender**.



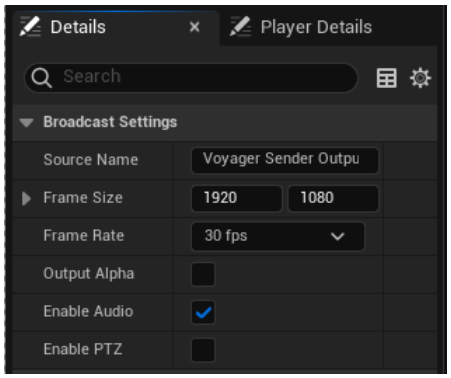
Create New NDI Media Sender

6. In the **Save Asset As** window, navigate to the folder where you want to save the asset and in the **Name** field, enter a name for the asset and select **Save**.

7. With the **NDI Media Sender** selected in the **Outliner**, in the **Details** tab, double-click the **NDI Media Sender** icon to open the editor.

Open NDI Media Sender Editor

The **NDI Media Sender** is populated automatically, but you can change the settings if necessary.



NDI Media Sender Editor - Details

8. When configured as desired, select **Save** and close the editor.

★Remove any unused NDI Sender assets from the project, as they are counted as NDI Streams licenses and will prevent you from adding new streams, if you have reached your license maximum.

Voyager Blueprints

In Voyager, you'll find a number of blueprints that automate actions, such as launching a level in a multi-level project, executing events, sending tally messages, using a DataLinq Multi-Value Actor, and scheduling Timecode events. The actions can be triggered in multiple ways, including by a Lucid Studio or RossTalk GPI Event, by a Voyager Trackless camera change, or using a DataLinq key.

The following blueprints are described in this section:

Creating a Blueprint to Launch Multiple Levels

If your project contains more than one level, you'll need to create a blueprint that will launch the level you want.

Multi-level projects present some challenges, beyond just launching the desired level. See the Unreal Engine documentation for more information.

The following procedures describe a simple blueprint that can be used to launch any of three levels. For the purpose of these instructions, the levels are named Studio_A, Studio_B, and Studio_C.

To create a new blueprint:

To create a variable to launch levels:

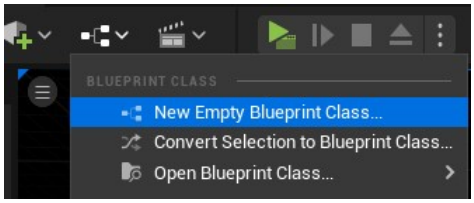
To load a specific level:

To unload a level:

To connect the nodes:

To create a new blueprint:

1. In the main toolbar, select the arrow beside the **Blueprints** icon and select **New Empty Blueprint Class**.



Select New Empty Blueprint Class

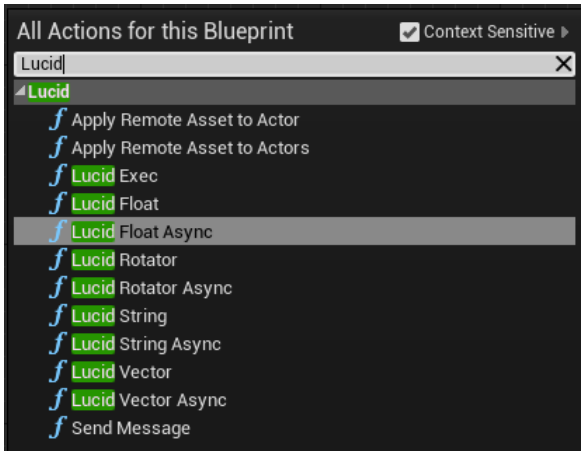
2. From the **Pick Parent Class** list, select any class.
3. In the **Create Blank Blueprint Class**, in the **Name** field, enter a name for your new blueprint and select **OK**.

   By default, the blueprint is saved in the **Blueprints** folder.

   The blueprint editor opens.

To create a variable to launch levels:

1. In the blueprint editor, select the **Event Graph** tab.
2. Right-click in the graph and in the **Search** field, type `Lucid`.

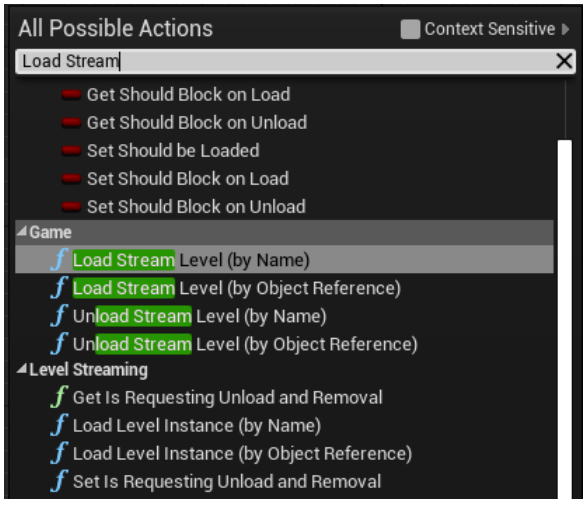


Search Results for Lucid Nodes

3. Select the **Lucid Exec** node.
4. In the **Lucid Exec** node, in the **Var Name** field, enter a name for the variable (e.g., Load_Levels).

To load a specific level:

1. Right-click in the graph and in the **Search** field, type `Load Stream`.

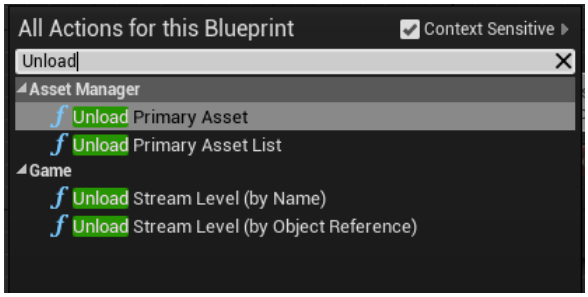


Search Results for Load Stream Level

2. Select **Load Stream Level** (by Name).
3. In the **Level Name** field, enter the name of the first level you want to load.

To unload a level:

1. Right-click in the graph and in the **Search** field, type `Unload Stream`.

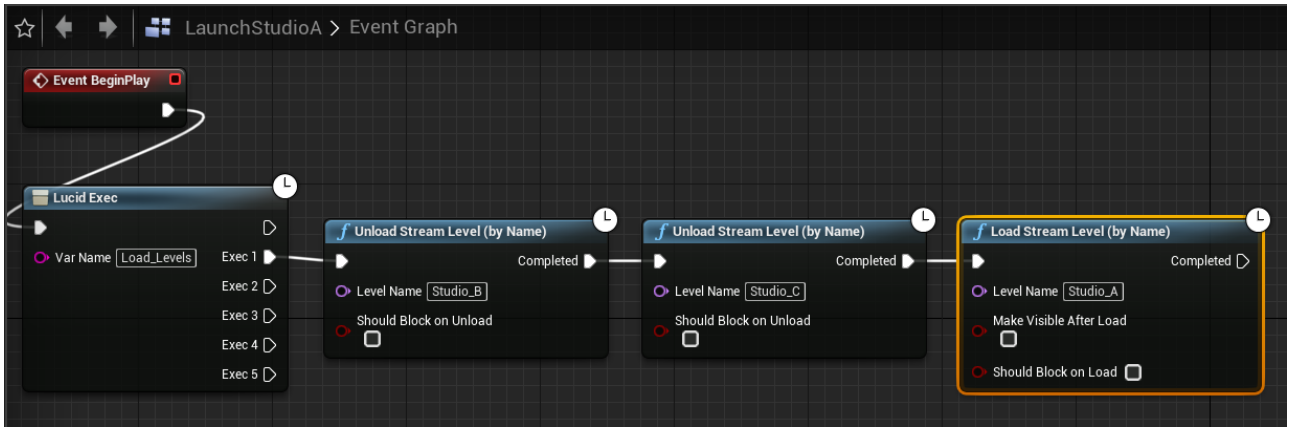


Search Results for Unload Stream Level

2. Select **Unload Stream Level** (by Name).
3. In the **Level Name** field, enter the name of the level that should not be launched.
4. Repeat Steps 2 and 3 to create additional **Unload Stream Level** nodes for the other levels in your project.

To connect the nodes:

1. Connect the **Output** pin of the **Event Begin Play** node to the Input pin of the **Lucid Exec** node.
2. Connect **Output** pin **Exec 1** of the **Lucid Exec** node to the **Input** pin of the first **Unload Stream Level** node (e.g., Studio_B).
3. Connect the **Output** pin of the **Unload Stream Level** node to the **Input** pin of the second **Unload Stream Level** node (e.g., Studio_C).
4. Connect the **Output** pin of the second **Unload Stream Level** node to the **Input** pin of the **Load Stream Level** node (e.g., Studio_A).



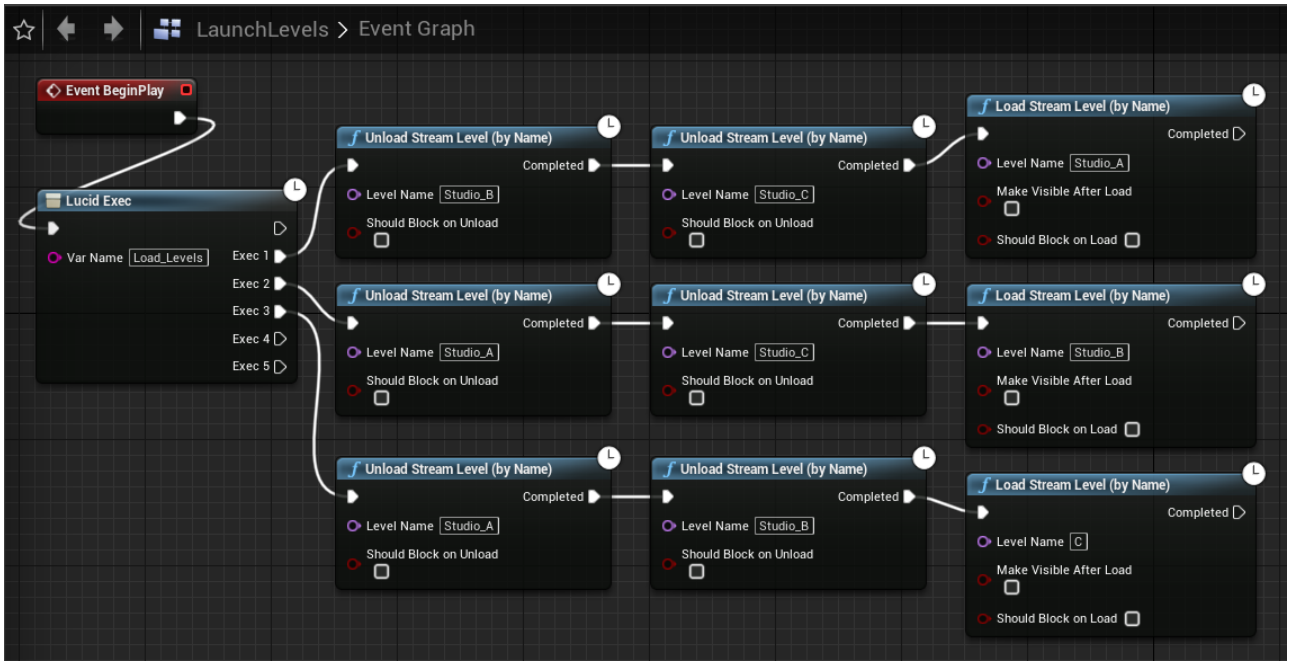
Blueprint to Launch Studio_A

5. Create additional **Unload Stream Level** and **Load Stream Level** nodes to launch Studio_B and Studio_C.

6. Connect the **Output** and **Input** pins in the same way as previously, starting with **Output** pin **Exec 2** in the **Lucid Node** and changing the names so that the **Load Stream Level** node will launch a different level.

Unload Stream Level	Load Stream Level
Studio_B, Studio_C	Studio_A
Studio_A, Studio_C	Studio_B
Studio_A, Studio_B	Studio_C

The blueprint for launching all three levels would look like this:



Blueprint to Launch Multiple Levels

7. Select **Save** and close the blueprint editor.

Voyager Event Execution

You can use **Voyager Event Execution** nodes to trigger lower thirds and animations, control camera presets, toggle object visibility, apply media dynamically, or trigger data updates.

Voyager Event Execution nodes register named events at **BeginPlay** that can be triggered externally by Lucid Studio events, Voyager Trackless events, RossTalk GPI, DashBoard, StreamDeck, or through Web Server control.

The following example describes how to use a **Voyager Event Execution** node to toggle visibility of a static mesh actor.

You will create two Voyager events:

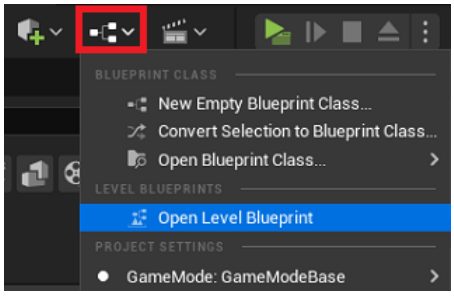
- **ShowLogo** — makes a mesh visible
- **HideLogo** — hides the mesh

Before you start, make sure you have the static mesh actor whose visibility you want to change placed in the level.

To register an event:

1. In Voyager, in the main toolbar, select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.

   The **Level Blueprint** is used because the static mesh actor is placed directly in the level.

   Open Level Blueprint

2. In the blueprint, in the **Event BeginPlay** node, drag off from the **Exec** pin and in the **Search** field, begin typing `sequence` and select the **Sequence** node.

   The **Sequence** node allows multiple **Voyager Event Execution** nodes to be initialized at **BeginPlay**.

3. In the **Sequence** node, do the following:

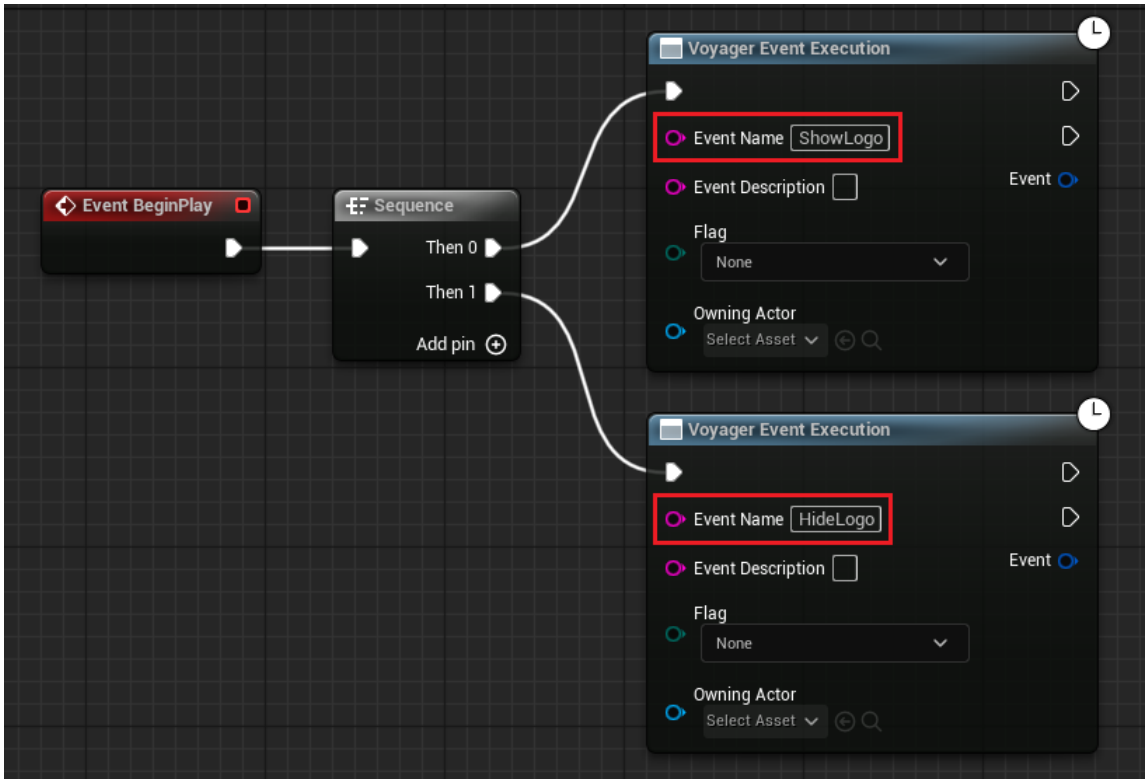
   - Drag off the **Then 0** pin and in the **Search** field, begin typing `Voyager` and select the **Voyager Event Execution** node.
   - Drag off the **Then 1** pin and in the **Search** field, begin typing `Voyager` and select the **Voyager Event Execution** node.

   You should now have two **Voyager Event Execution** nodes connected to the **Sequence** node.

4. In each **Voyager Event Execution** node, in the **Event Name** field, enter a name for the event, e.g., ShowLogo and HideLogo.

   These names will appear in the external control system (Voyager Trackless, Lucid Studio, etc.)

This is how the blueprint should look at this point:



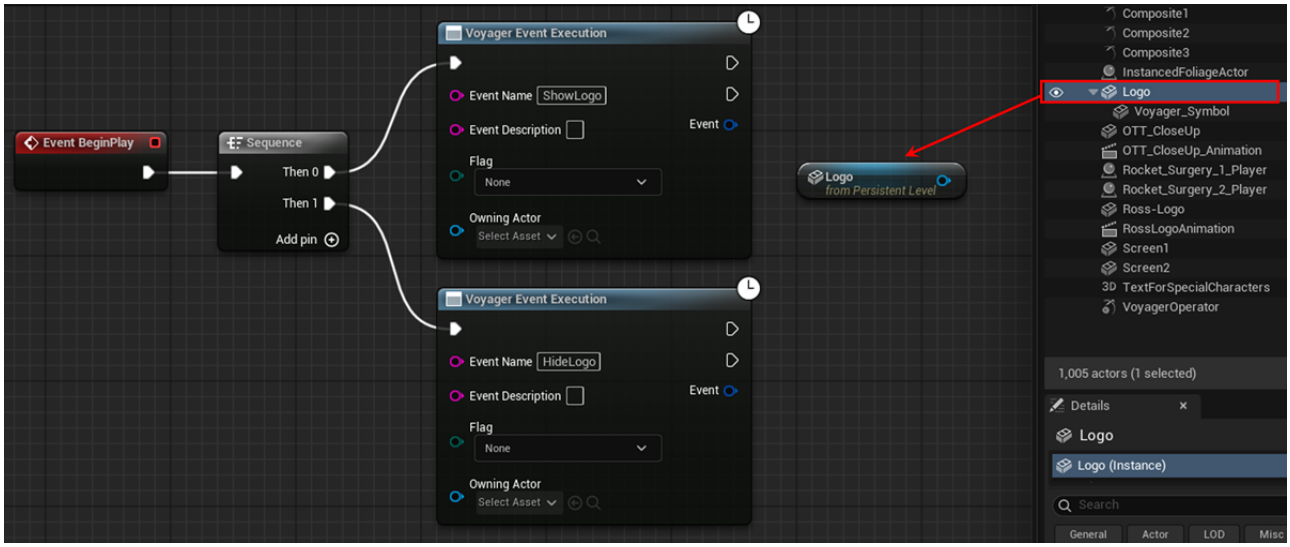
Voyager Event Execution Blueprint - Beginning

To reference the target animation:

1. Make sure that Voyager is in **Editor** mode, not **PIE** mode.
2. From the **Outliner** panel, drag the static mesh actor you want to toggle visibility on, into the blueprint.

   This adds a **Reference** node for the actor to the blueprint, telling it which object to control.

   ★If the referenced static mesh actor is removed from the level, the blueprint will generate errors.



Voyager Event Execution Blueprint - Add Reference

3. In the **Reference** node, do the following:
    - Drag out from the blue pin and in the **Search** field, begin typing `Set Visibility` and select the **Set Visibility (StaticMeshComponent)** node.
    - Repeat to add a second **Set Visibility** node.

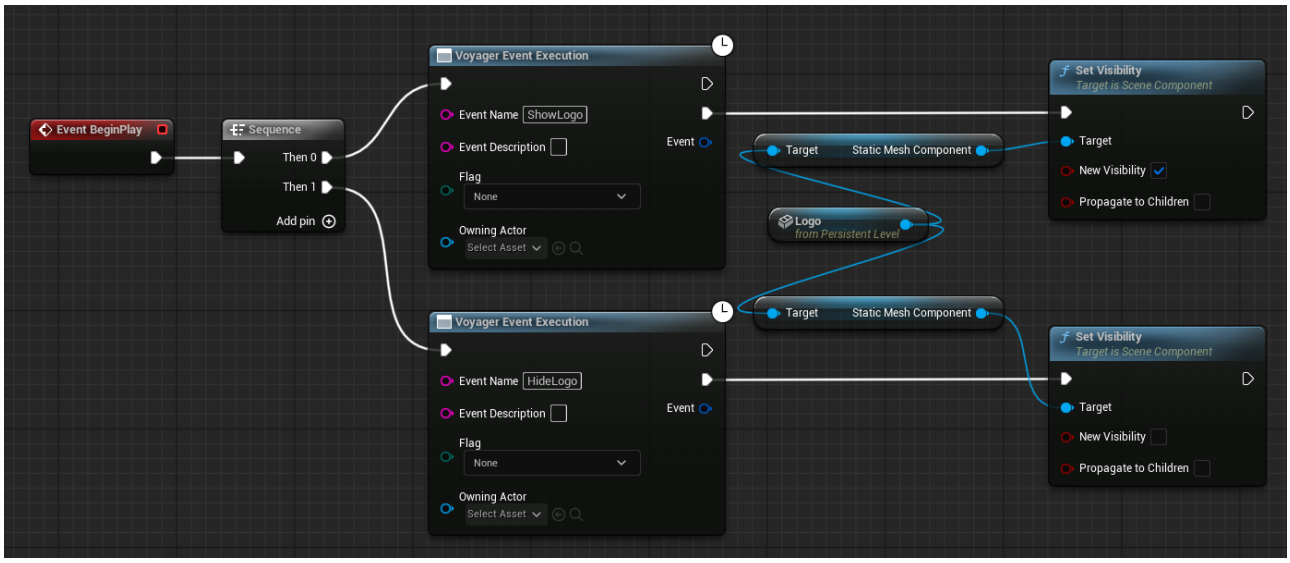
To connect playback logic:

1. Make the following connections:
   - Connect the **Event output** execution pin of the **Voyager Event Execution** (ShowLogo) node to the **Exec** input pin of the first **Set Visibility** node.
   - Connect the **Event output** execution pin of the **Voyager Event Execution** (HideLogo) node to the **Exec** input pin of the second **Set Visibility** node.
2. In the first **Set Visibility** node, select the **New Visibility** checkbox.

   This makes the object visible.

   The second node must have the checkbox unchecked. This hides the object.

   The blueprint should look like this:



Voyager Event Execution Blueprint

3. Select **Compile** and then select **Save**.

   When the level is played, the events are registered and appear in the control system (Voyager Trackless, Lucid Studio, etc.). They must then be triggered by the control system to be executed.

   When **ShowLogo** is triggered, the mesh becomes visible.

   When **HideLogo** is triggered, the mesh is hidden.

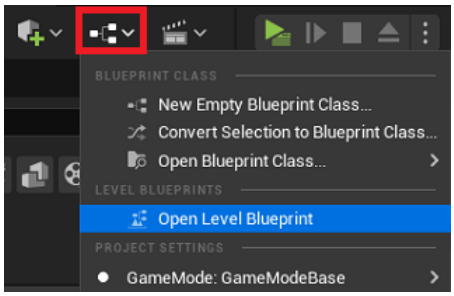
Voyager Event Execution with Delay

Voyager Event Execution with Delay nodes function like standard Voyager Event Execution nodes but add a frame-based delay before executing.

The following example describes how to use a **Voyager Event Execution with Delay** node to play a level sequence animation with a delay of a number of frames.

To register a delayed event:

1. In Voyager, in the main toolbar, select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.



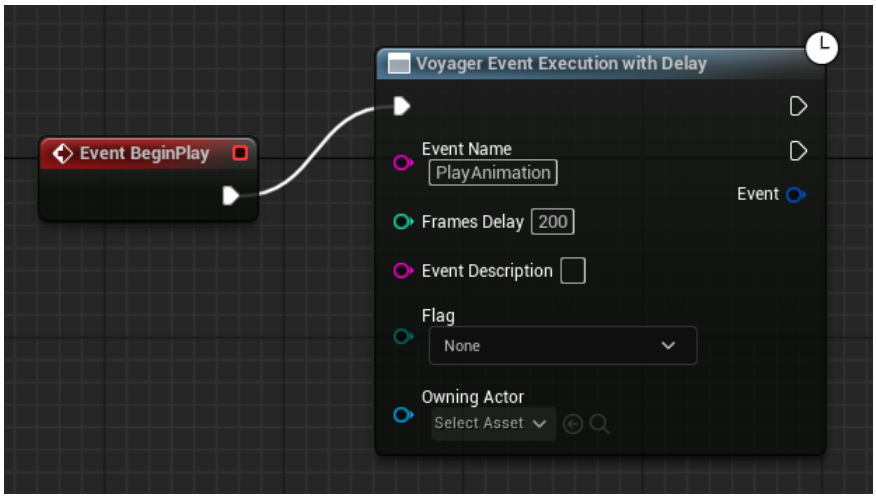
Open Level Blueprint

2. In the blueprint, in the **Event BeginPlay** node, drag off from the **Exec** pin and in the **Search** field, begin typing `Voyager` and select the **Voyager Event Execution with Delay** node.
3. In the **Voyager Event Execution with Delay** node, do the following:
   - In the **Event Name** field, enter a name for the event, e.g., PlayAnimation.

     These names will appear in the external control system (Voyager Trackless, Lucid Studio, etc.)
   - In the **Frames Delay** field, enter the number of frames to wait before playing the animation.

     The delay is measured in frames and is dependent on the project's frame rate (for example, 200 frames at 30 fps equals approximately 6.67 seconds).

   This is how the blueprint should look at this point:

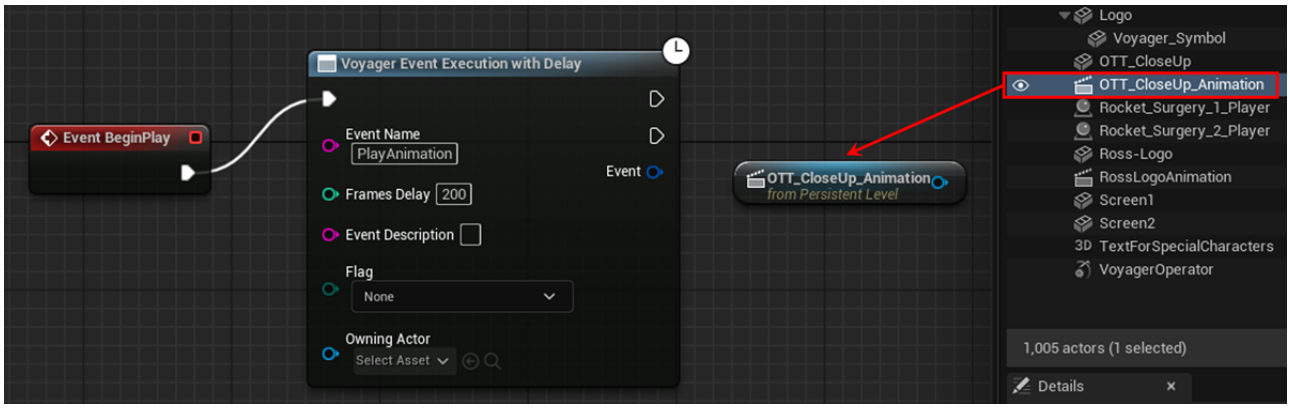


Voyager Event Execution with Delay - Beginning

To reference the target animation:

1. Make sure that Voyager is in **Editor** mode, not **PIE** mode.
2. From the **Outliner** panel, drag the level sequence actor containing the animation, into the blueprint.

   This adds a **Reference** node for the actor to the blueprint, telling it which object to control.



Voyager Event Execution with Delay - Add Reference

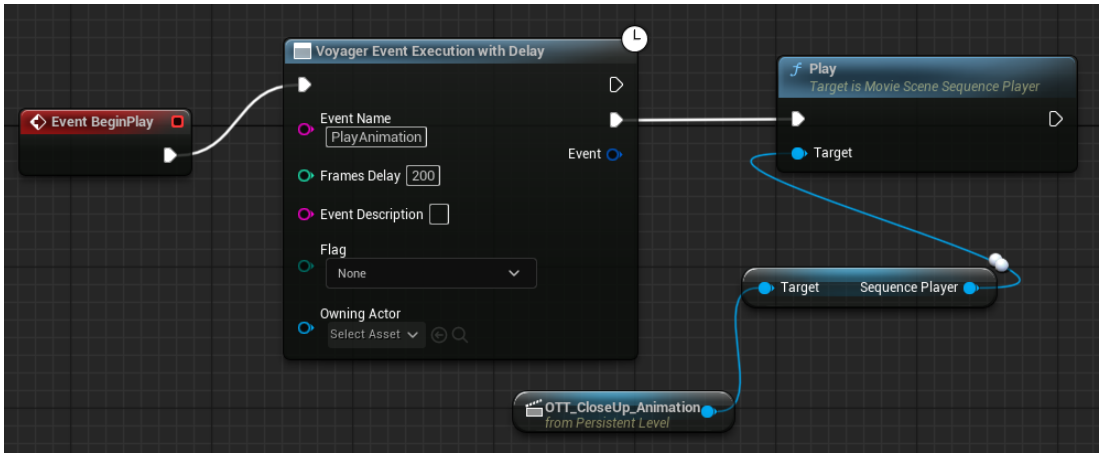
3. In the level sequence actor's **Details**, under **Playback**, ensure that **Auto Play** is disabled to allow manual control.
4. In the **Reference** node, drag out from the blue pin and in the **Search** field, begin typing `Get Sequence Player` and select the **Get Sequence Player** node.
4. In the **Get Sequence Player** node, drag out from the blue pin and in the **Search** field, begin typing `Play` and select the **Play** node.

   The **Play** node controls the **Level Sequence** through its **Sequence Player**.

To connect playback logic:

1. Connect the **Event output** execution pin of the **Voyager Event Execution** with **Delay** node to the **Exec** input pin of the **Play** node.

   The blueprint should look like this:



Voyager Event Execution with Delay Blueprint

2. Select **Compile** and then select **Save**.

When the level is played, the event is registered and appears in the control system (Voyager Trackless, Lucid Studio, etc.). It must then be triggered by the control system to be executed.

Tally Event

You can use the **Tally Event** node to receive tally signals from switchers, routers, or tally systems and trigger an action, such as turning on a tally light.

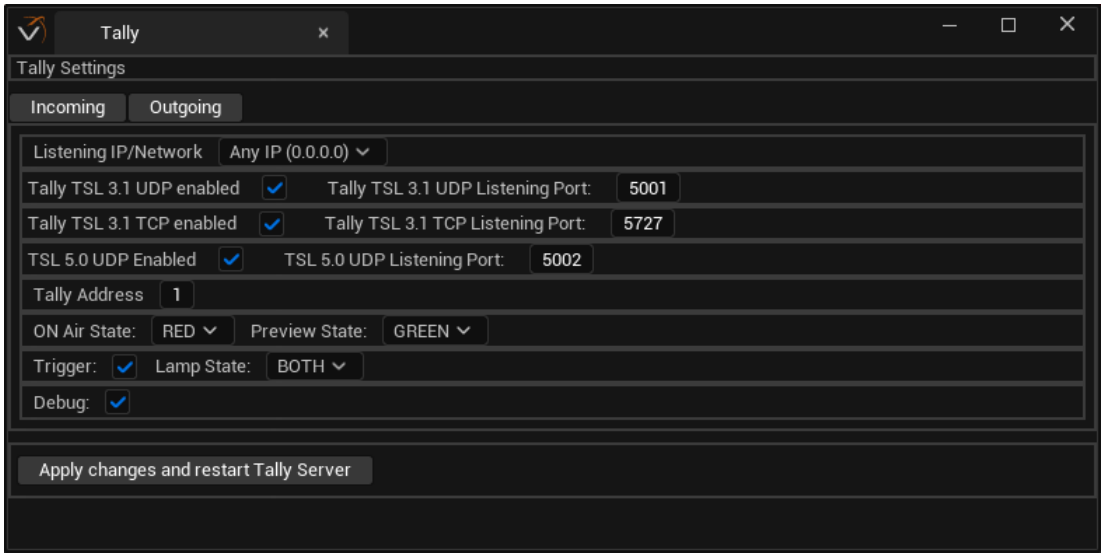
Before you start, ensure that:

- Voyager is connected to the production switcher.
- The Tally Plugin is enabled (see Enabling the Voyager Plugins).
- The switcher, router, or tally system supports tally messaging.
- The incoming tally message settings are configured in Voyager.

To configure incoming tally settings:

1. In Voyager, select **Window > Voyager > Tally Settings**.

   The **Tally Settings** window opens.



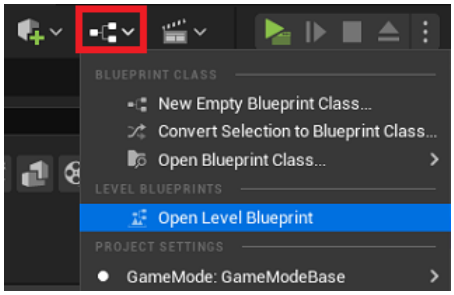
2. Select the **Incoming** tab (typically selected by default).
3. In the **Listening IP/Network** field, enter the IP/network address on which the switcher, router, or tally system will connect or select it from the drop-down, if it's been configured previously.

   Optionally, you can keep the default address of **Any IP**.
4. Select/deselect the other settings as necessary.
5. Select **Apply changes and restart Tally Server** to activate the new settings.

To build the blueprint:

1. In Voyager, add a point light actor into the level and name it **TallyLight**.

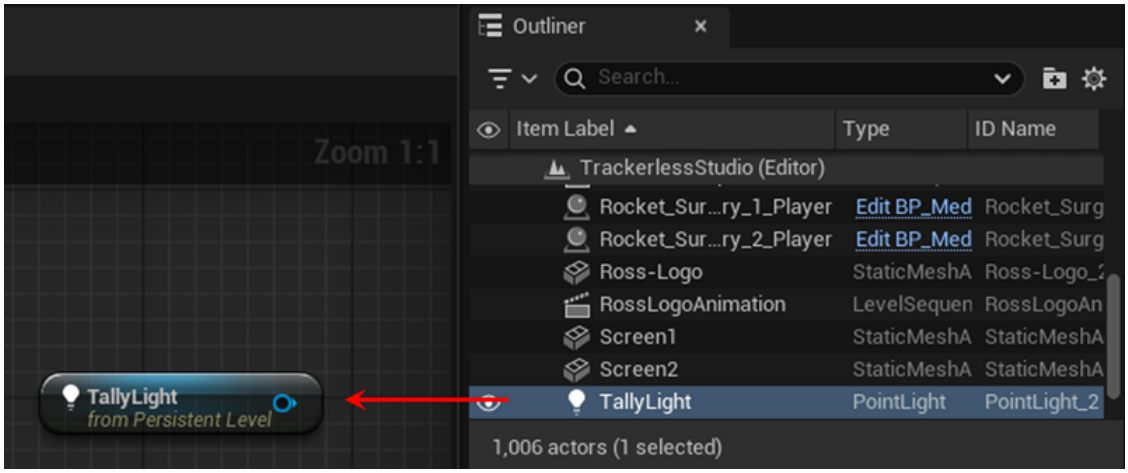
2. Then, in the main toolbar, select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.



Open Level Blueprint

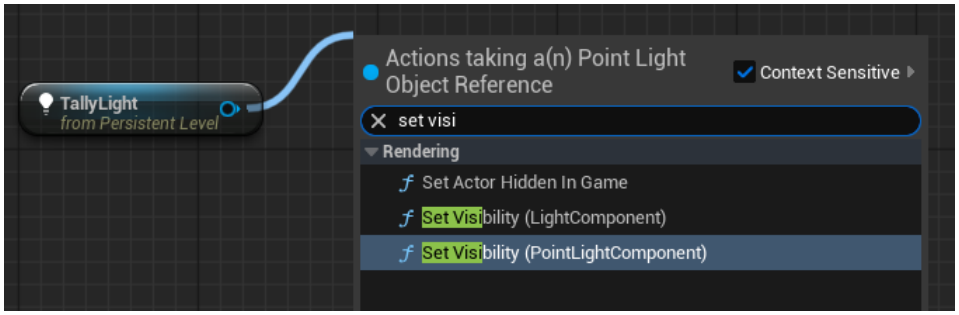
3. Drag the **TallyLight** actor from the **Outliner** into the blueprint.

   This adds a reference node for the actor to the blueprint, telling it which object to control.



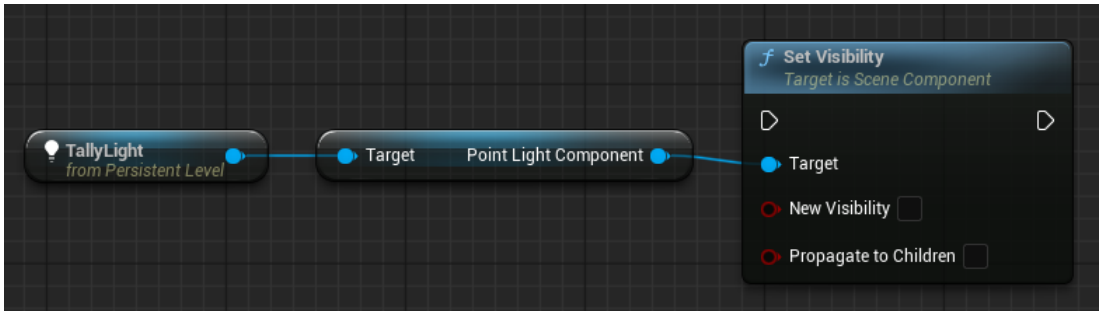
Drag TallyLight into the Blueprint

4. Drag off from the blue pin on the **TallyLight** reference node, begin typing `Set Visibility` and select the **Set Visibility (PointLightComponent)** node.



Add Set Visibility Node

This adds the **Set Visibility** node and the connecting **Point Light Component** node.



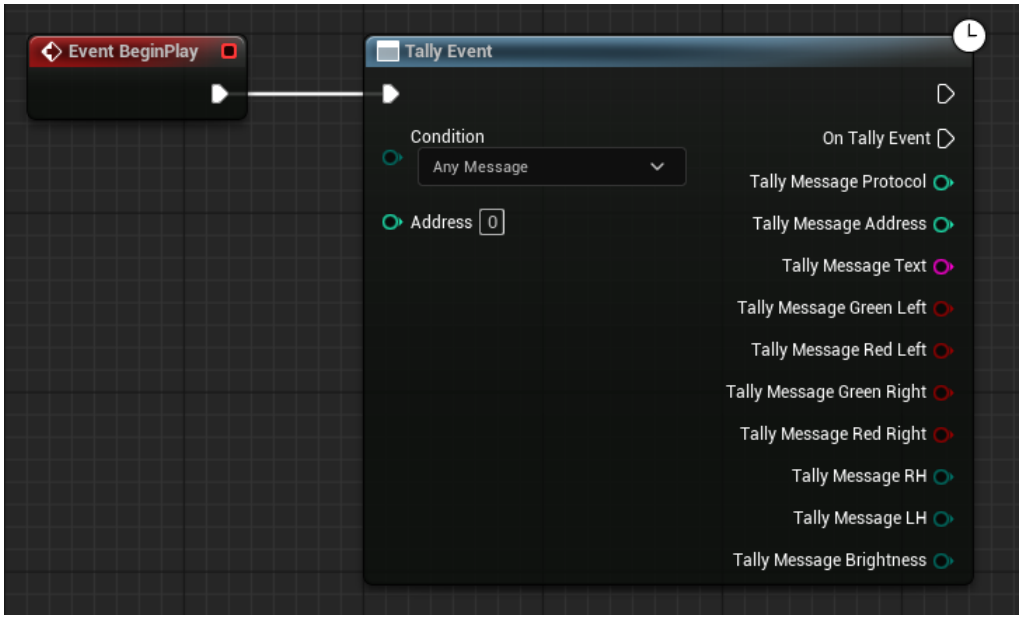
Set Visibility Node Added

5. Drag off from the **Point Light Component** node, begin typing `Set Visibility` and add a second **Set Visibility** node.

   This will add a second **Point Light Component** node as well, but this can be deleted. Only one **Point Light Component** node is necessary.

6. Drag off from the **Event Begin Play** node, begin typing `Tally Event` and select the **Tally Event** node.

7. Right click on the **Tally Message** pin and select **Split Struct Pin** to expose the **Tally** options.



Tally Event with Tally Message Pin Expanded

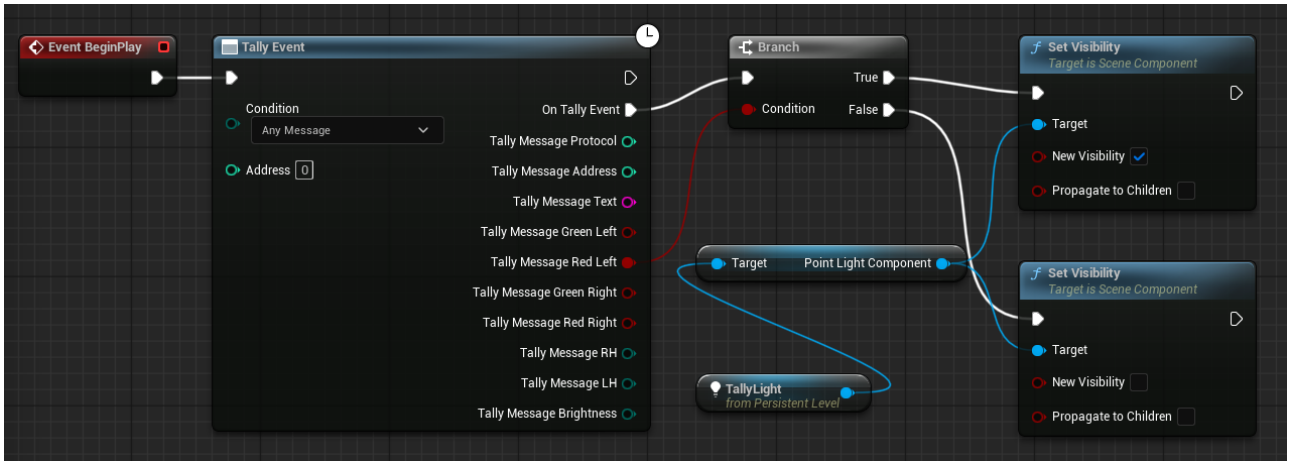
8. Drag off from the **On Tally Event** pin on the **Tally Event** node and add a **Branch** node.

To connect playback logic:

1. Make the following connections:
   - Connect the **True** pin on the **Branch** node to the **Exec** pin of the first **Set Visibility** node.
   - Connect the **False** pin on the **Branch** node to the **Exec** pin on the second **Set Visibility** node.
   - Connect the **Tally Message Red Left** pin on the **Tally Event** node to the **Condition** pin on the **Branch** node.

2. Select the **New Visibility** checkbox on the **Set Visibility** node that is connected to the **True** pin of the **Branch** node.
3. Leave the **New Visibility** checkbox on the **Set Visibility** node that is connected to the **False** pin of the **Branch** node clear.

   The completed blueprint should resemble the following example.



4. Select **Compile** and then select **Save**.

   When **Tally Message Red Left** is true, the tally light is turned on. When it is false, the tally light is turned off.

Send Tally Message from Voyager Trackless

You can use the **Send Tally Message** to send tally signals from Voyager to a production switcher when the active camera changes in Voyager Trackless.

The switcher uses this information to trigger the tally light on the active camera, allowing on-camera talent and operators to see which camera is live.

Before you start, ensure that:

- Voyager is connected to the production switcher.
- The Tally Plugin is enabled (see Enabling the Voyager Plugins).
- The switcher supports tally messaging.
- The outgoing tally message settings are configured in Voyager.

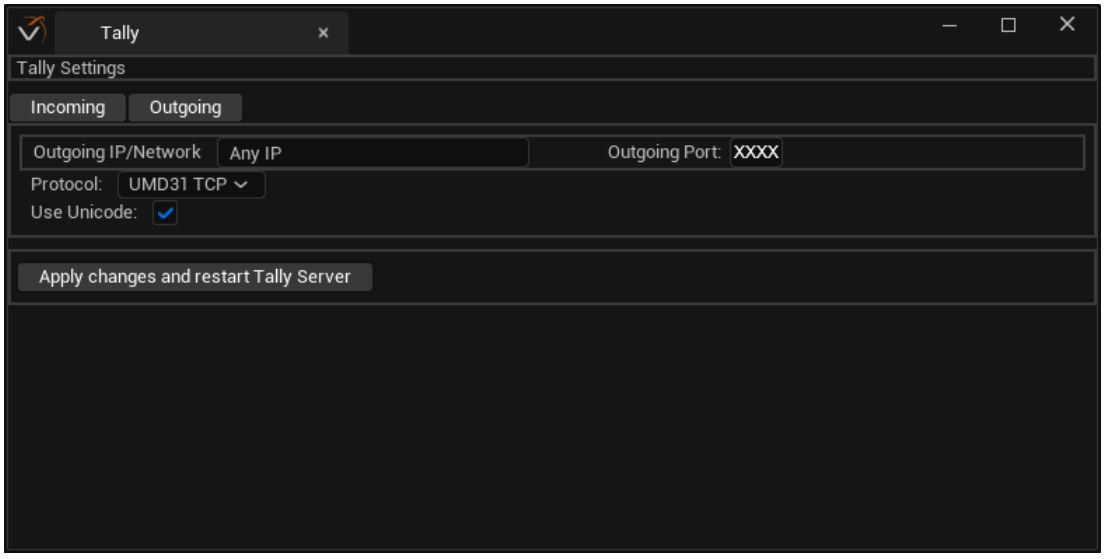
Workflow:

1. Configure the outgoing tally message settings.
2. Set up the camera mapping.
3. Build the blueprint.

To configure outgoing tally settings:

1. In Voyager, select **Window > Voyager > Tally Settings**.

   The **Tally Settings** window opens.



   Tally Settings

2. In the **Tally Settings** window, select the **Outgoing** tab.
3. In the **Outgoing** tab, do the following:
   - In the **Outgoing IP/Network** field, enter the IP/Network address of the switcher.
   - In the **Outgoing Port** field, enter the port number on which to communicate with the switcher.
   - From the **Protocol** drop-down, select the protocol used by the switcher.

The options are:

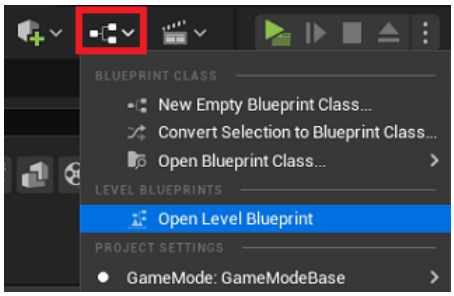
- **UMD31 TCP** — TCP-based TSL protocol for UMD displays and tally systems
- **UMD31 UDP** — UDP-based TSL protocol, often used by multiviewers
- **UMD 50** — Extended TSL protocol supporting additional features

- Select the **Use Unicode** checkbox, if necessary.

4. Select **Apply changes and restart Tally Server** to activate the new settings.

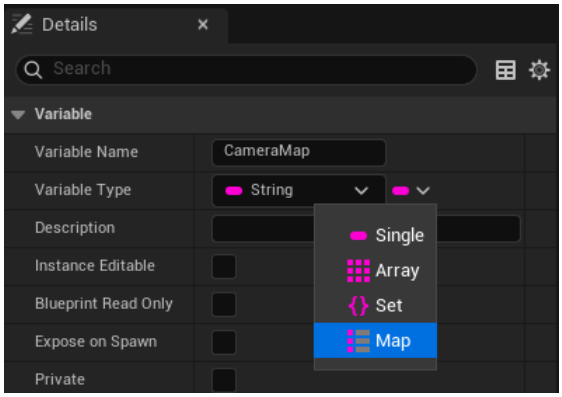
To set up camera mapping:

1. In Voyager, in the main toolbar, select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.



Open Level Blueprint

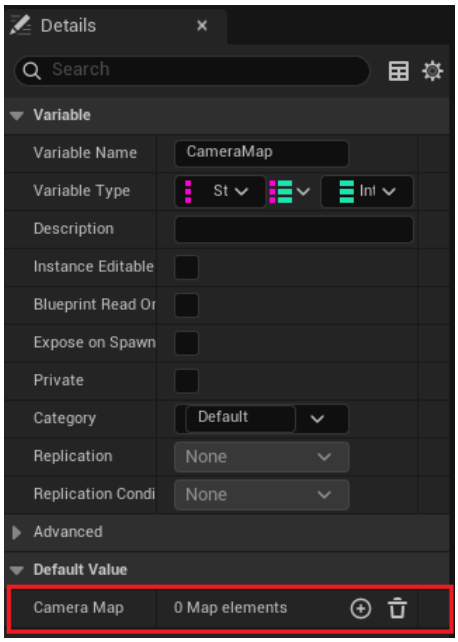
2. In the blueprint, in the **My Blueprint** panel to the left of the blueprint, select the **+** icon to add a variable.
3. Name the variable **CameraMap** and make it a **String** type.
4. With the **CameraMap** variable selected, in the **Details** panel to the right of the blueprint, from the **Container Type** drop-down adjacent to the **Variable Type**, select **Map**.



Select Map Container Type

5. Select **Compile**.

   The **CameraMap** table is added at the bottom of the Details tab.



CameraMap Table

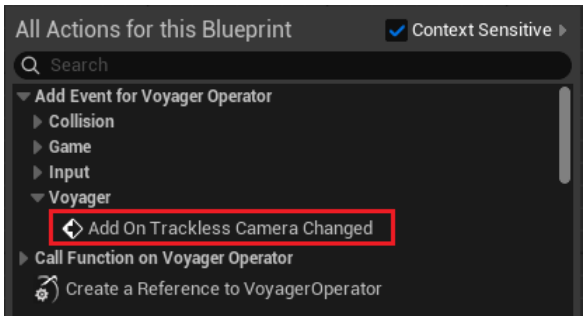
6. In the **CameraMap** table, select the **+** icon to add a map element.
7. In the **Key** field, enter the name of one of the cameras in the level and in the **Value** field, enter the index of the camera.
8. Repeat steps 6 and 7 to add a row for each camera to the table.

   The completed table should resemble the following example.

CameraMap Table Complete

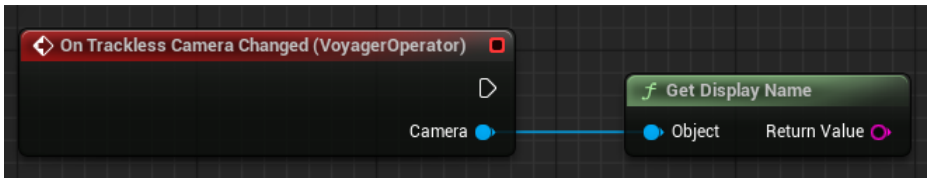
To build the blueprint:

1. With the **VoyagerOperator** selected in the **Outliner** panel, right-click in an empty area of the blueprint and select **Add Event for Voyager Operator > Voyager > Add On Trackless Camera Changed**.



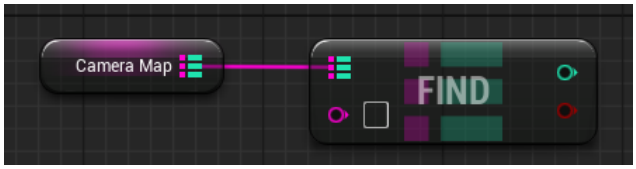
Add On Trackless Camera Changed

2. Drag off the **Camera** pin of the **On Trackless Camera Changed** node, begin typing `Get Display Name` and select the **Get Display Name** node.



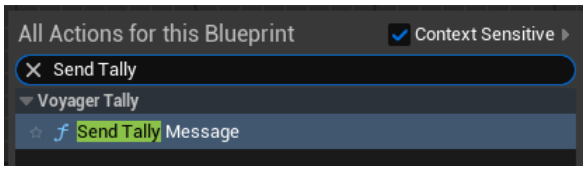
Add Get Display Name Node

3. Drag the **CameraMap** variable into the blueprint, selecting **Get CameraMap**.
4. Drag off from the right side of the **CameraMap** variable, begin typing `Find` and select the **Find** node.



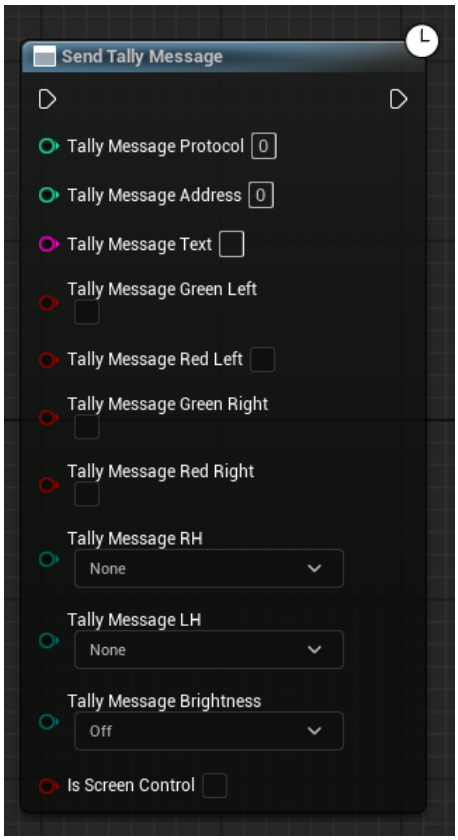
Add Find Node to CameraMap Variable

5. Right-click in the blueprint, begin typing `Send Tally Message` and select the **Send Tally Message** node.



Select Send Tally Message

6. In the **Send Tally Message** node, right-click **Tally Message** and select **Split Struct Pin** to expand the pin.



Send Tally Message Node

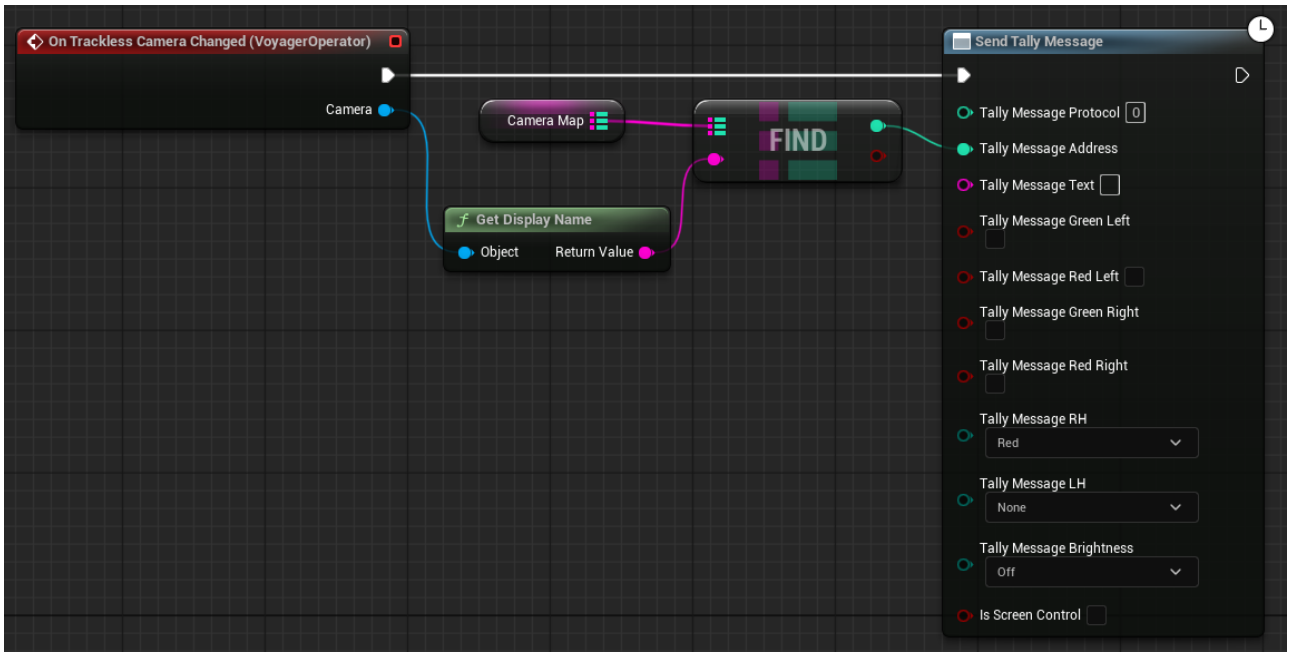
7. In the **Send Tally Message** node, select an option for the type of tally message to send (see table below).

Field	Description	Typical Use
Tally Message Protocol	Specifies the communication protocol used to send the tally message (e.g., TSL, serial, network-based tally).	Ensures the switcher sends tally data in a format that the receiving device understands.
Tally Message Address	Defines the destination address or ID of the device/UMD display that should receive the tally message.	Used to route tally information to the correct tally device.
Tally Message Text	Specifies the text label included in the tally message.	Typically used to send the source name (e.g., CAM1, VTR2, CG) to a UMD display.
Tally Message Green Left	Turns on the green indicator on the left side of the tally display.	Usually used to indicate Preview status.
Tally Message Red Left	Turns on the red indicator on the left side of the tally display.	Usually used to indicate Program (on-air) status.
Tally Message Green Right	Turns on the green indicator on the right side of the tally display.	Often used for secondary preview or status from another bus/ME.
Tally Message Red Right	Turns on the red indicator on the right side of the tally display.	Often used for program status from another bus/ME.
Tally Message RH	Controls the right-hand tally indicator as a general on/off indicator.	Used by devices that treat the right tally as a single indicator instead of separate colors.
Tally Message LH	Controls the left-hand tally indicator as a general on/off indicator.	Used when the device expects a single left indicator rather than separate red/green signals.
Tally Message Brightness	Sets the brightness level of the tally or display indicator.	Adjusts visibility of the tally lights or UMD display.
Is Screen Control	Determines whether the message controls the display screen itself rather than just the tally lights.	Used when the message needs to control screen behavior (display text, brightness, or screen state).

To connect playback logic:

1. Make the following connections:
   - Connect the **Exec** pin on the **On Trackless Camera Changed** node to the **Exec** input pin of the **Send Tally Message** node.
   - Connect the green pin on the **Find** node to the **Tally Message Address** pin on the **Send Tally Message** node.
   - Connect the **Return Value** pin on the **Get Display Name** node to the pink **Key** pin on the **Find** node.

   The completed blueprint should resemble the following example.



Send Tally Message Blueprint

2. Select **Compile** and then select **Save**.

   When the level is played and Voyager receives a message from Voyager Trackless that the active camera has changed, Voyager sends the message to the production switcher to activate the tally light on the active camera. In this example, the tally light will turn red.

   ★If the **Find** node returns an error, check that the camera names in the **CameraMap** variable exactly match the camera names in the level.

To verify the setup:

1. Play the level.
2. In Voyager Trackless, in the **Production Control Panel**, switch cameras.
3. Confirm that the correct tally light activates on the switcher.

Send Tally Message from Lucid Studio

You can use the **Send Tally Message** blueprint node to send a tally signal from a Lucid Studio event to a production switcher, for example, to let the switcher operator know that a graphic is ready to go on air, or to trigger an action.

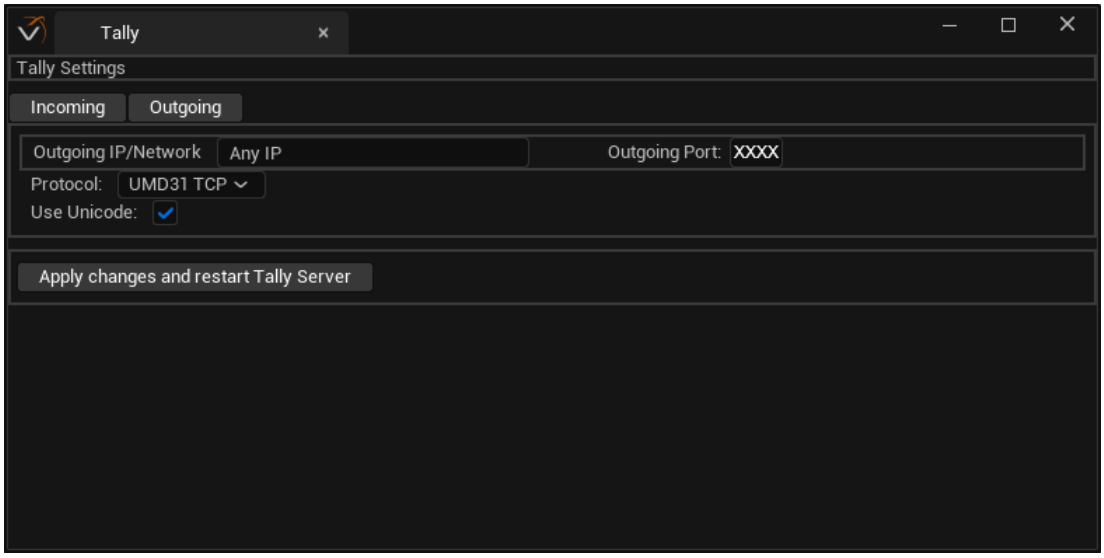
Before you start, ensure that:

- Voyager is connected to the production switcher.
- The switcher supports tally messaging.
- The Tally Plugin is enabled (see Enabling the Voyager Plugins).
- An event has been created in Lucid Studio to trigger the tally message.
- The outgoing tally message settings are configured in Voyager.

To configure outgoing tally settings:

1. In Voyager, select **Window > Voyager > Tally Settings**.

   The **Tally Settings** window opens.



   Tally Settings

2. In the **Tally Settings** window, select the **Outgoing** tab.
3. In the **Outgoing** tab, do the following:
   - In the **Outgoing IP/Network** field, enter the IP/Network address of the switcher.
   - In the **Outgoing Port** field, enter the port number on which to communicate with the switcher.
   - From the **Protocol** drop-down, select the protocol used by the switcher.

     The options are:

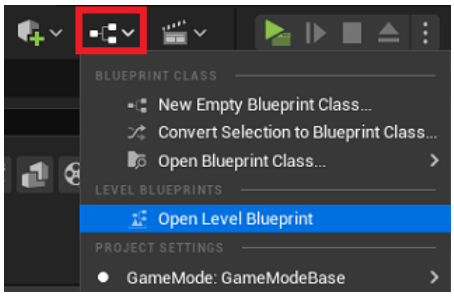
     - UMD31 TCP — TCP-based TSL protocol for UMD displays and tally systems
     - UMD31 UDP — UDP-based TSL protocol, often used by multiviewers
     - UMD 50 — Extended TSL protocol supporting additional features

- Select the **Use Unicode** checkbox, if necessary.

4. Select **Apply changes and restart Tally Server** to activate the new settings.

To create a blueprint to send a tally message:

1. In Voyager, in the main toolbar, select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.



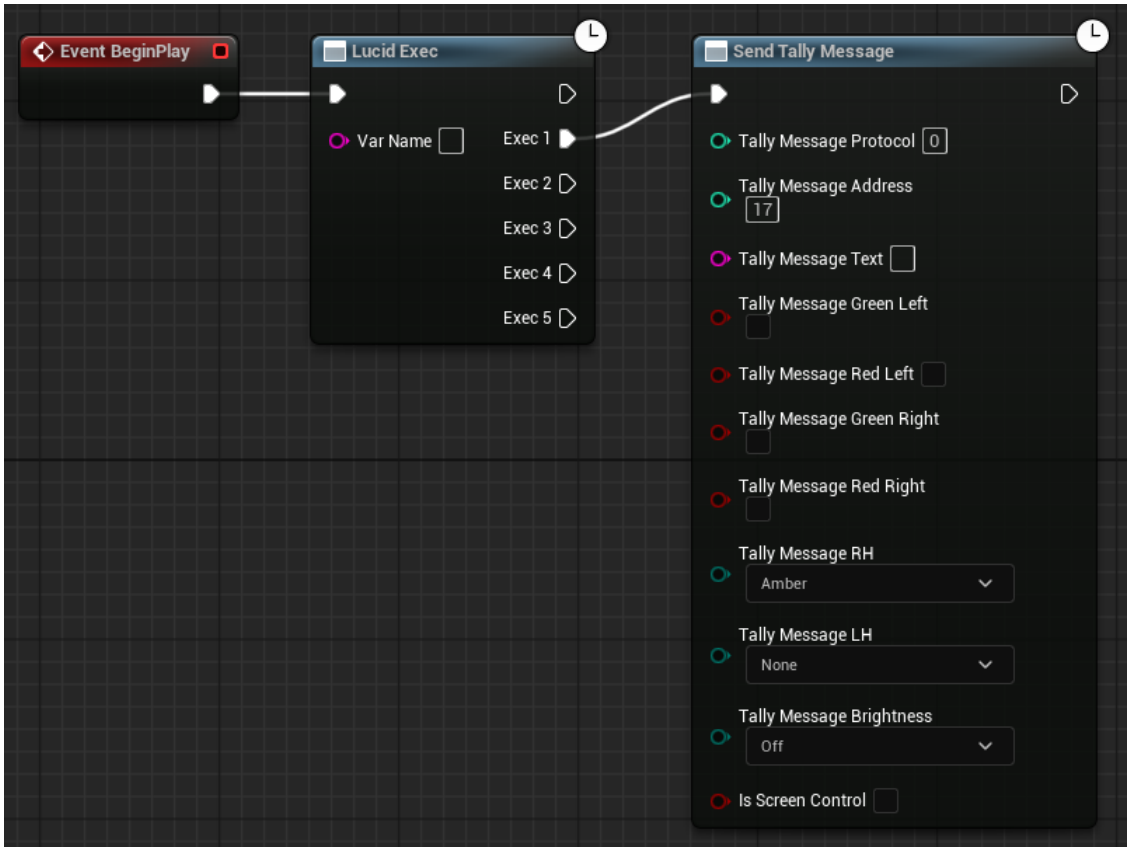
Open Level Blueprint

2. In the blueprint, in the **Event BeginPlay** node, drag off from the **Exec** pin and in the **Search** field, begin typing `Lucid` and select the **Lucid Exec** node.
3. In the **Lucid Exec** node, drag off from the **Exec 1** pin and in the **Search** field, begin typing **Send Tally Message** and select the **Send Tally Message** node.
4. In the **Send Tally Message** node, do the following:
   - In the **Tally Message Address** pin, enter the index of the switcher to which the message should be sent.
   - Select an option for the type of tally message to send (see table below).

Field	Description	Typical Use
Tally Message Protocol	Specifies the communication protocol used to send the tally message (e.g., TSL, serial, network-based tally).	Ensures the switcher sends tally data in a format that the receiving device understands.
Tally Message Address	Defines the destination address or ID of the device/UMD display that should receive the tally message.	Used to route tally information to the correct tally device.
Tally Message Text	Specifies the text label included in the tally message.	Typically used to send the source name (e.g., CAM1, VTR2, CG) to a UMD display.
Tally Message Green Left	Turns on the green indicator on the left side of the tally display.	Usually used to indicate Preview status.
Tally Message Red Left	Turns on the red indicator on the left side of the tally display.	Usually used to indicate Program (on-air) status.
Tally Message Green Right	Turns on the green indicator on the right side of the tally display.	Often used for secondary preview or status from another bus/ME.
Tally Message Red Right	Turns on the red indicator on the right side of the tally display.	Often used for program status from another bus/ME.
Tally Message RH	Controls the right-hand tally indicator as a general on/off indicator.	Used by devices that treat the right tally as a single indicator instead of separate colors.

Field	Description	Typical Use
Tally Message LH	Controls the left-hand tally indicator as a general on/off indicator.	Used when the device expects a single left indicator rather than separate red/green signals.
Tally Message Brightness	Sets the brightness level of the tally or display indicator.	Adjusts visibility of the tally lights or UMD display.
Is Screen Control	Determines whether the message controls the display screen itself rather than just the tally lights.	Used when the message needs to control screen behavior (display text, brightness, or screen state).

The completed blueprint should resemble the following example.



Send Tally Message from Lucid Studio Blueprint

5. Select **Compile** and then select **Save**.

   The example blueprint would flash an amber light on the switcher, indicating that a graphic is ready to go on air.

Using the DataLinq Multi-Value Actor in a Blueprint

When dealing with large amounts of data, you will probably want to set up the functionality in a blueprint. The DataLinq Multi-Value Actor makes it easy to do this. You'll need to select your DataLinq source and then configure the DataLinq Multi-Value actor in the blueprint.

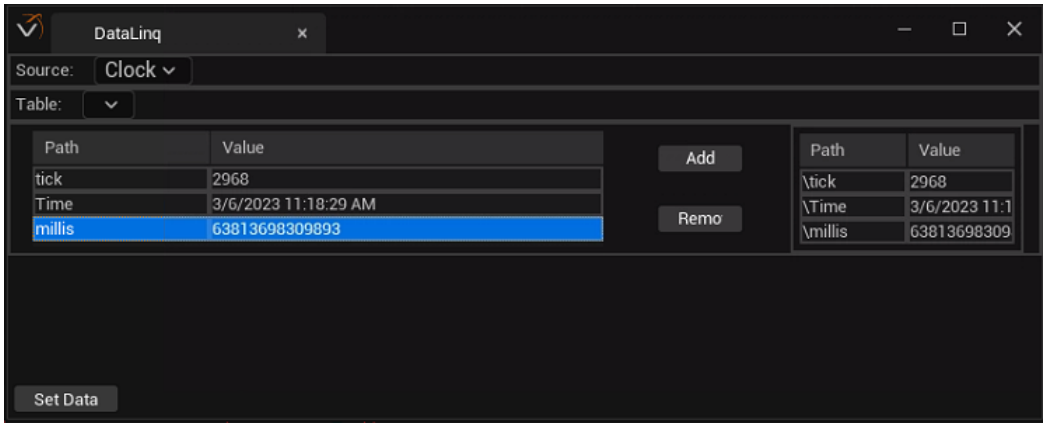
Before you start, ensure that:

The DataLinq Plugin is enabled and configured (see Configuring the DataLinq Plugin).

To select the DataLinq source:

1. In the **Place Actors** tab, from the **DataLinq** category, drag the **DataLinq MultiValue Actor** into the level.

   The DataLinq source window opens.

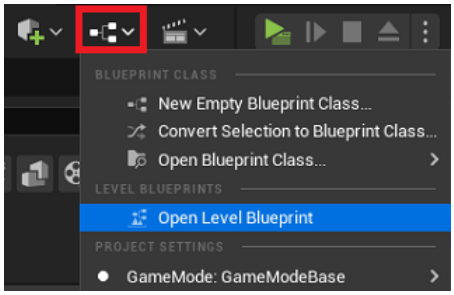


DataLinq - Select DataLinq Source

2. From the **Source** drop-down, select the DataLinq source you want to use.
3. If you have more than one table, from the **Table** drop-down, select the table you want to use.
4. Then double-click cells in the table to select the rows and columns you want to display in your project.
5. Select **Set Data**.

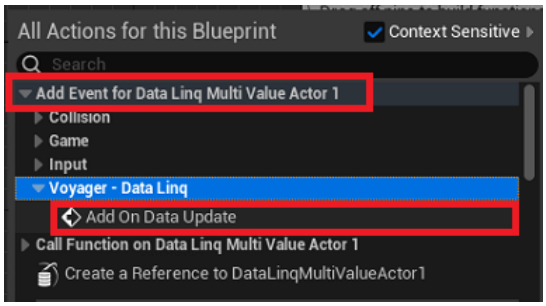
To configure the DataLinq Multi-Value actor in the blueprint:

1. In the **Outliner** select the **DataLinq Multi-Value Actor** you added.
2. Then select **Blueprints > Open Level Blueprint** and right-click in an empty area of the blueprint.



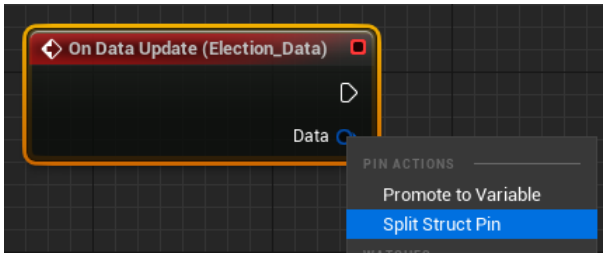
DataLinq - Open Level Blueprint

3. Expand **Add Event for DataLinq MultiValue Actor** and then expand **Voyager - DataLinq** and select **Add On Data Update**.



DataLinq Blueprint - Add On Data Update

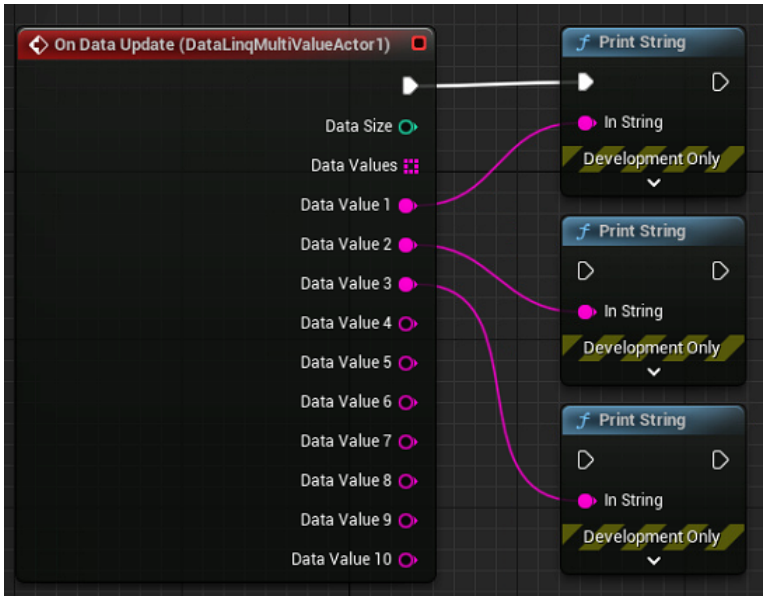
4. In the **On Data Update** node, right-click the **Data** pin and select **Split Struct Pin**.



DataLinq Blueprint - Split Struct Pin

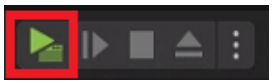
5. Then connect as many pins as needed to the desired output node.

Example:



DataLinq Blueprint - Connect Output Nodes

6. Select **Compile** and **Save** and close the blueprint.
7. Select **Play** to check that the data is being successfully retrieved.



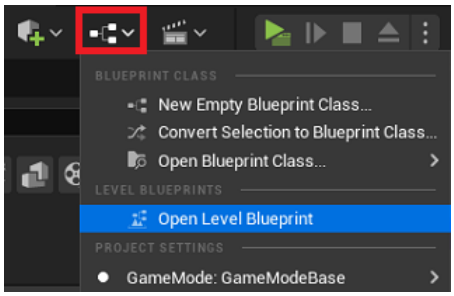
Play Button

Setting up Timecode Events in the Level Blueprint

Once you have configured the Timecode Provider, you can then use the Timecode to schedule and execute events in the level blueprint. You can schedule the execution of events based on a specified hour, minute, second, and frame.

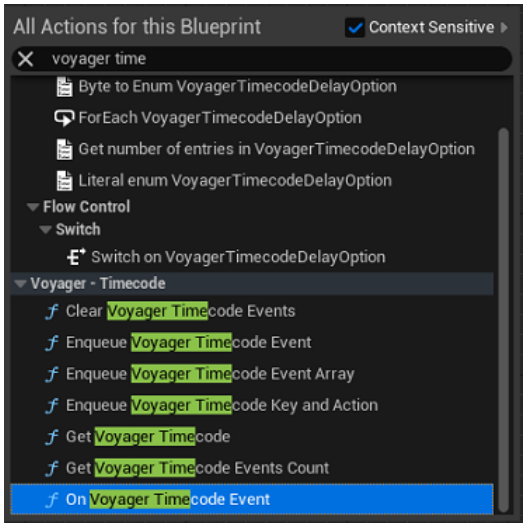
To access Voyager Timecode Blueprint nodes:

1. In Voyager, in the main toolbar, select the arrow beside the **Blueprints** icon and select **Open Level Blueprint**.



Open Level Blueprint

2. Inside the level blueprint, right-click and in the **All Actions for this Blueprint** field, begin typing `Voyager - Timecode` to search for the **Voyager - Timecode** category.



Voyager - Timecode Category

3. From the list, select one of the following blueprint nodes and continue with the instructions for that node:
   - Enqueue Voyager Timecode Event
   - Enqueue Voyager Timecode Array
   - Enqueue Voyager Timecode Key and Action
   - On Voyager Timecode Events
   - Clear Voyager Timecode Events
   - Get Voyager Timecode
   - Get Voyager Timecode Events Count

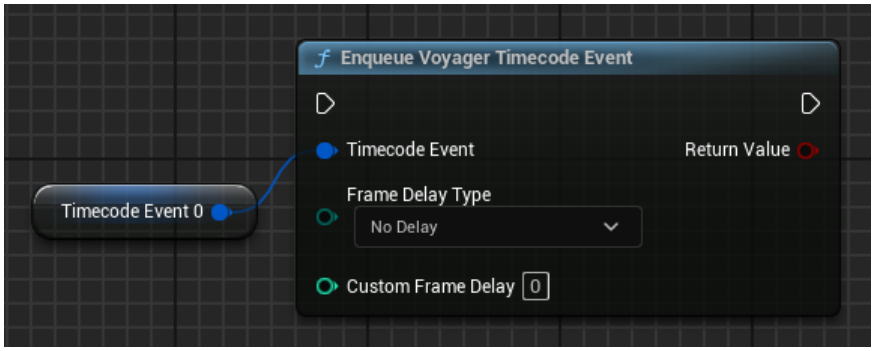
Enqueue Voyager Timecode Event

Use the **Enqueue Voyager Timecode Event** node to schedule a Voyager timecode-based event. The event will be triggered when the configured Timecode is reached.

To enqueue a Voyager Timecode event:

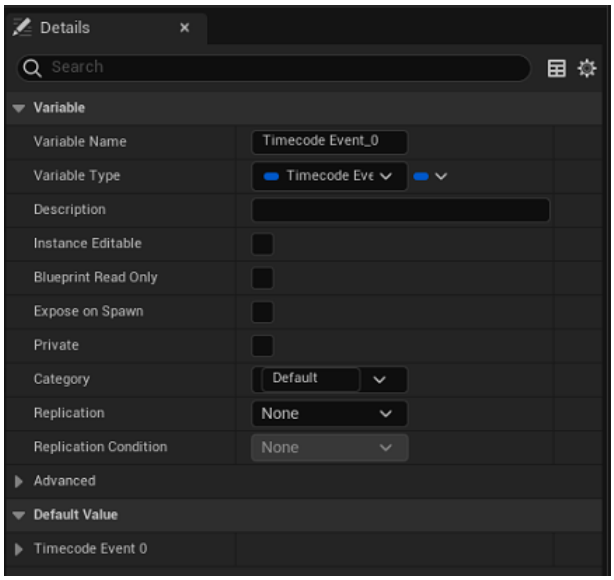
1. In the **Enqueue Voyager Timecode Event** node, right-click the **Timecode Event** pin and from the menu, select **Promote to Variable**.

   The **Timecode Event** variable node is added to the **Enqueue Voyager Timecode Event** node.



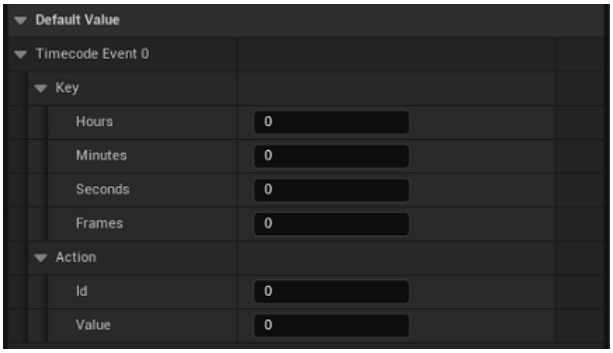
Enqueue Voyager Timecode Event - Promote to Variable

2. Compile the blueprint and then select the **Timecode Event** variable node to access the **Details** panel.



Enqueue Voyager Timecode Event - Details

3. Expand **Timecode Event** and then expand **Key** and **Action**.

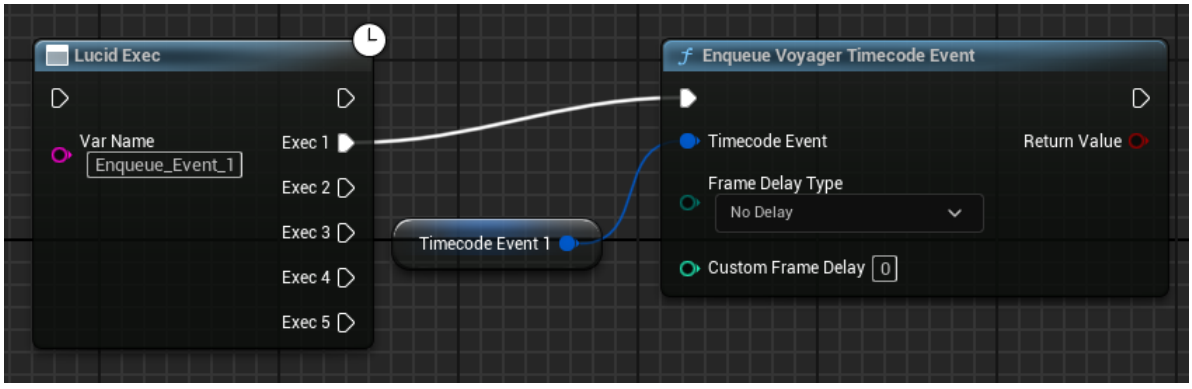


Enqueue Voyager Timecode Event - Timecode Event Details

4. In the **Key** section, in the **Hours**, **Minutes**, **Seconds**, and **Frames** (optional) fields, enter the time that the event should occur.
5. In the **Action** section, enter the **Id** that represents the event and the **Value** of the actor associated with the event.

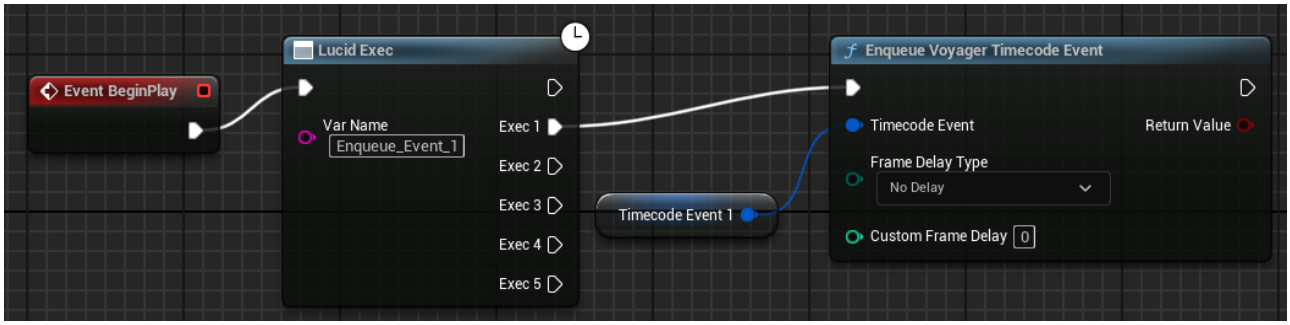
   For example, if the event is a camera switch, then the **Id** can be any number you want and the **Value** would be the number representing the specific camera that will be impacted by the event.
6. From the **Frame Delay Type** drop-down, select one of the following options:
   - **No Delay** — the event will occur immediately when the signal is received from the event node.
   - **General Settings Delay** — the event will occur following the frame delay set in the General Settings dialog, when the signal is received from the event node.
   - **Custom Delay** — the event will occur following the frame delay set in the **Custom Frame Delay** field in this node, when the signal is received from the event node.
7. Add an input node (e.g., RossTalk GPI or Lucid Exec) and connect the execution pin to the **Input** pin of the **Enqueue Voyager Timecode Event** node.

   When Voyager receives the signal from the input node, it will enqueue the event.



Enqueue Voyager Timecode Event - From Lucid Exec Node

8. Add the **Event BeginPlay** node to the input node.



Enqueue Voyager Timecode Event - Add Event Begin Play Node

9. Select **Save** and then **Compile**.

   The event is put into the queue of scheduled events.

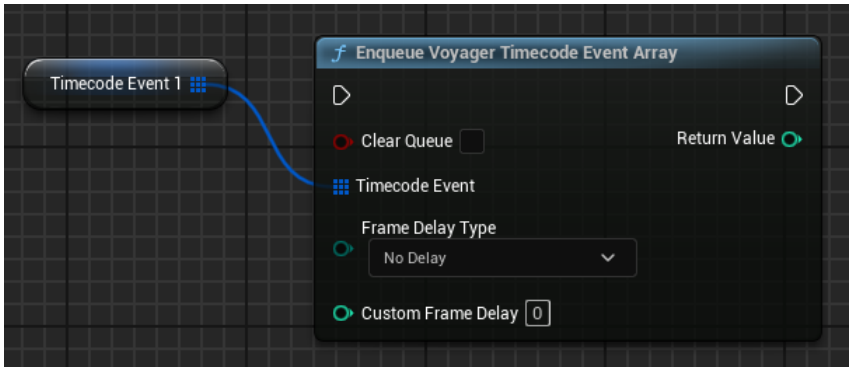
Enqueue Voyager Timecode Event Array

Use the **Enqueue Voyager Timecode Array** node to add multiple new Voyager timecode-based events to the queue of scheduled events. The events will be triggered when the configured Timecodes are reached.

To enqueue a Voyager Timecode Event Array:

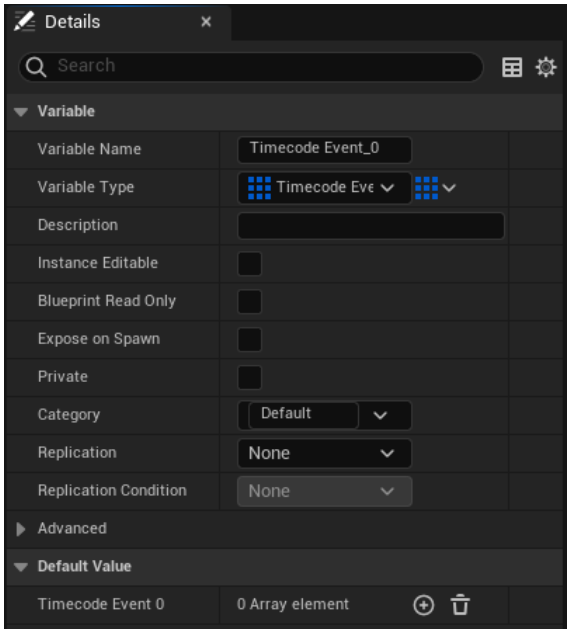
1. In the **Enqueue Voyager Timecode Event Array** node, right-click the **Timecode Event** pin and from the menu, select **Promote to Variable**.

   The **Timecode Event** variable node is added to the **Enqueue Voyager Timecode Event** node.



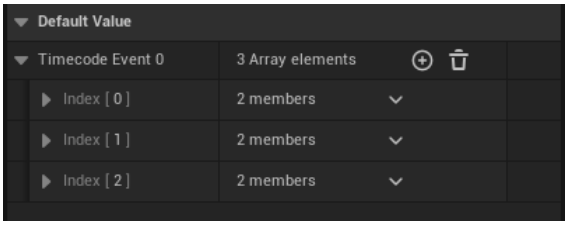
Enqueue Voyager Timecode Event Array - Promote to Variable

2. Select the **Clear Queue** checkbox to ensure that the current queue is completely clear before scheduling further timecode events.
3. Compile the blueprint.
4. Select the **Timecode Event** variable node to access the **Details** panel.



Enqueue Voyager Timecode Event Array - Details

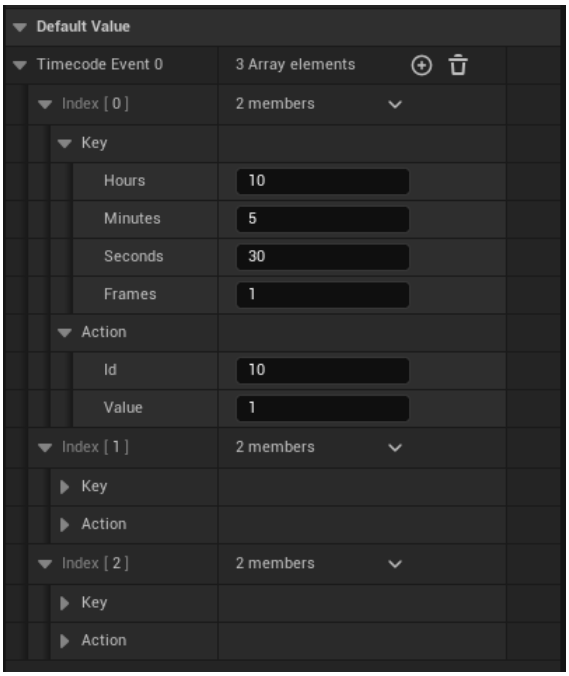
5. Expand the **Default Value** section and select the + icon beside **Array element** to add as many Array elements as you have Timecode events.



Enqueue Voyager Timecode Event Array - Add Array Elements

6. Expand each **Index** and then expand each **Key** and **Action** section and do the following:

   - In the **Key** section, in the **Hours**, **Minutes**, **Seconds**, and **Frames** (optional) fields, enter the time that the event should occur.
   - In the **Action** section, enter the **Id** that represents the event and the **Value** of the actor associated with the event.



Enqueue Voyager Timecode Event Array - Event 1 Details

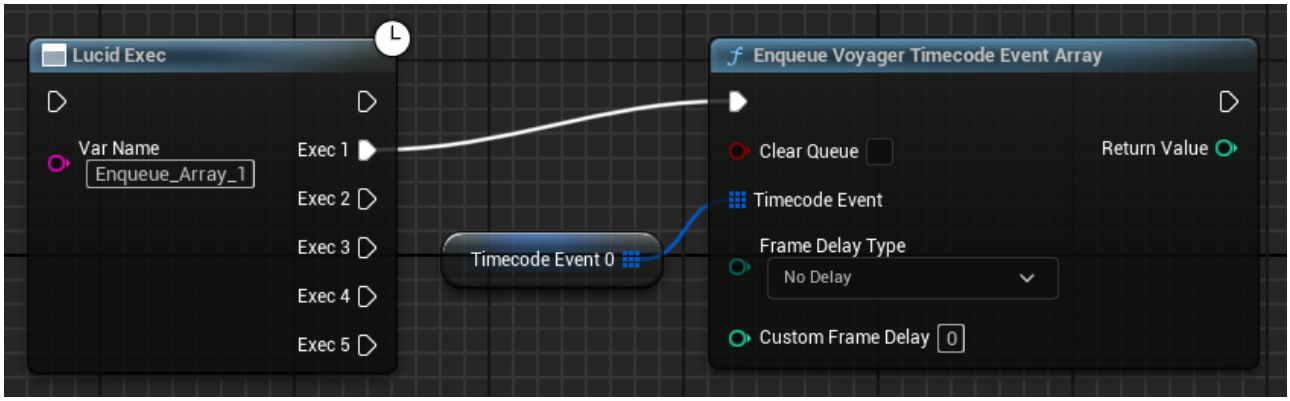
   For example, if the event is a camera switch, then the **Id** can be any number you want and the **Value** would be the number representing the specific camera that will be impacted by the event.

7. From the **Frame Delay Type** drop-down, select one of the following options:

   - **No Delay** — the event will occur immediately when the signal is received from the event node.
   - **General Settings Delay** — the event will occur following the frame delay set in the General Settings dialog, when the signal is received from the event node.
   - **Custom Delay** — the event will occur following the frame delay set in the **Custom Frame Delay** field in this node, when the signal is received from the event node.

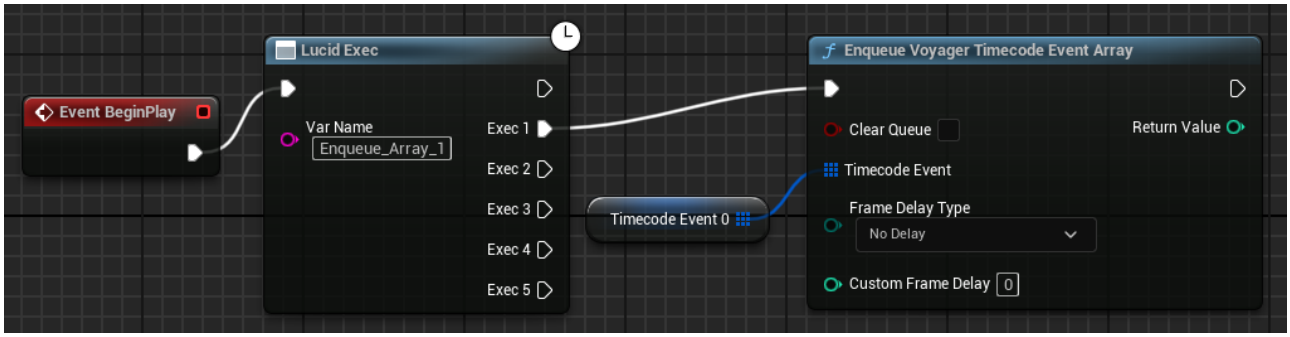
8. Add an input node (e.g., RossTalk GPI or Lucid Exec) and connect the execution pin to the **Input** pin of the **Enqueue Voyager Timecode Event Array** node.

   When Voyager receives the signal from the input node, it will enqueue the events.



Enqueue Voyager Timecode Event Array - From Lucid Exec Node

9. Add the **Event BeginPlay** to the input node.



EEnqueue Voyager Timecode Event Array - Add Event Begin Play Node

10. Select **Save** and then **Compile**.

    The events are put into the queue of scheduled events.

Enqueue Voyager Timecode Key and Action

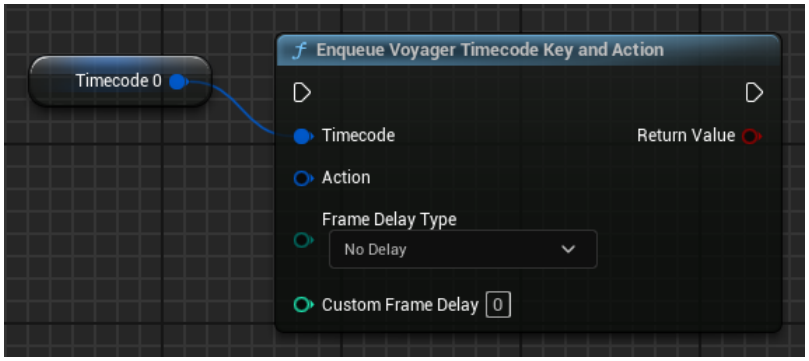
Use the **Enqueue Voyager Timecode Key and Action** node to add a new Voyager timecode-based event to the queue that includes a separate **Timecode Event Key** and **Action**.

Specify the timecode and Action separately from each other.

To enqueue Voyager Timecode Key and Action:

1. In the **Enqueue Voyager Timecode Key and Action** node, right-click the **Timecode** pin and from the menu, select **Promote to Variable**.

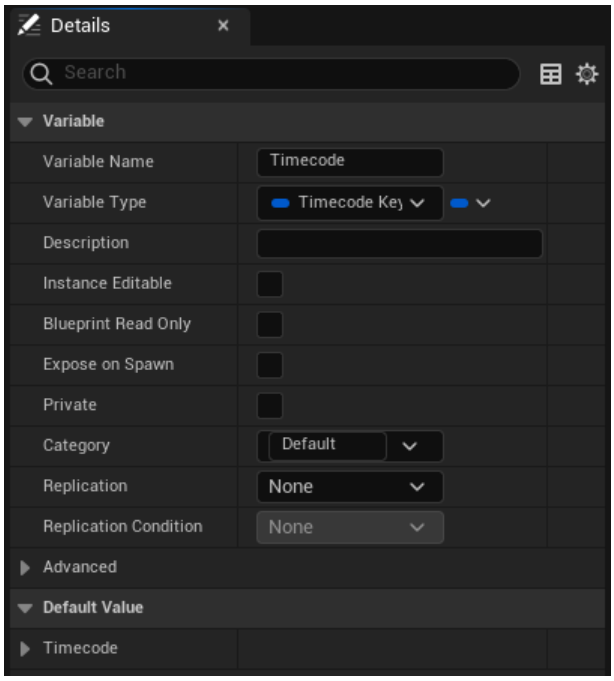
   The **Timecode** variable node is added to the **Enqueue Voyager Timecode Key and Action** node.



Enqueue Voyager Timecode Key and Action - Promote Timecode to Variable

2. Select **Compile**.

   The **Details** panel opens

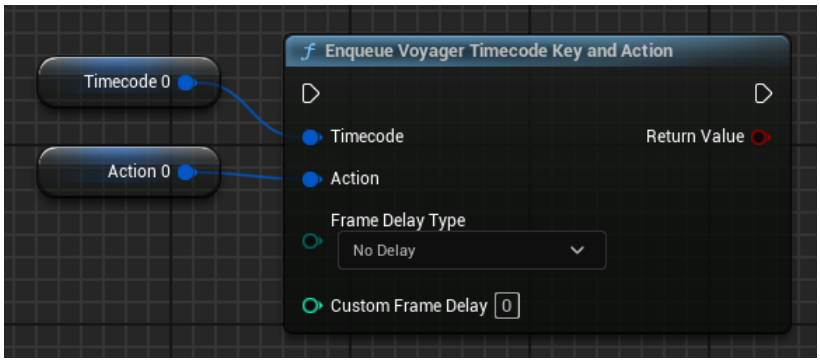


Enqueue Voyager Timecode Key and Action - Details

3. Select the **Timecode** node and in the **Details** panel, expand **Timecode**, and in the **Hours**, **Minutes**, **Seconds**, and **Frames** (optional) fields, enter the time that the event should occur.

Enqueue Voyager Timecode Key and Action - Timecode Details

4. Right-click the **Action** pin, and from the menu, select **Promote to Variable**.

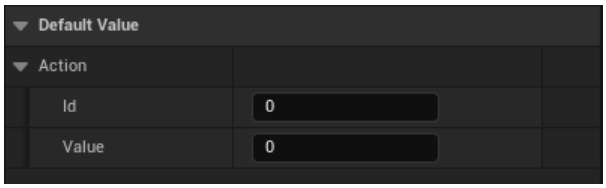


Enqueue Voyager Timecode Key and Action - Promote Action to Variable

5. Select **Compile**.

6. Select the **Action** node and in the **Details** panel, expand the **Action** section and enter the **Id** that represents the event and the **Value** of the actor associated with the event.

   For example, if the event is a camera switch, then the **Id** can be any number you want and the **Value** would be the number representing the specific camera that will be impacted by the event.



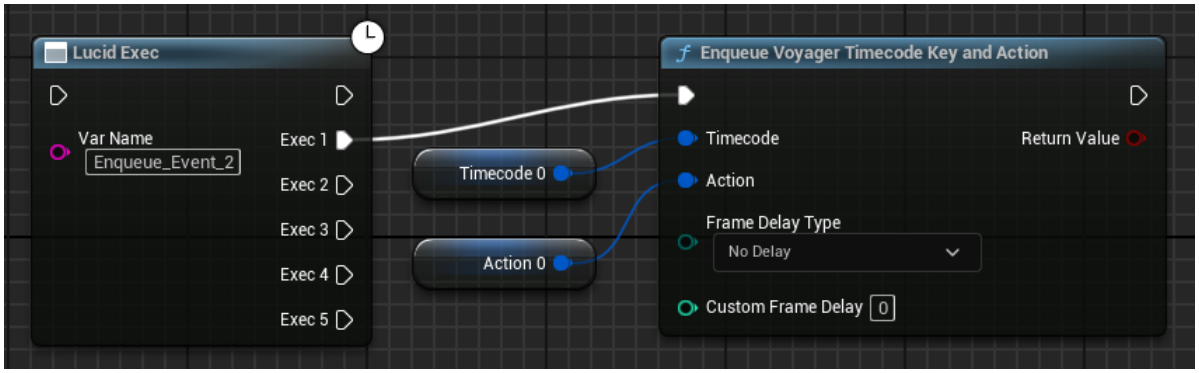
Enqueue Voyager Timecode Key and Action - Action Details

7. From the **Frame Delay Type** drop-down, select one of the following options:

   - **No Delay** — the event will occur immediately when the signal is received from the event node.
   - **General Settings Delay** — the event will occur following the frame delay set in the General Settings dialog, when the signal is received from the event node.
   - **Custom Delay** — the event will occur following the frame delay set in the **Custom Frame Delay** field in this node, when the signal is received from the event node.

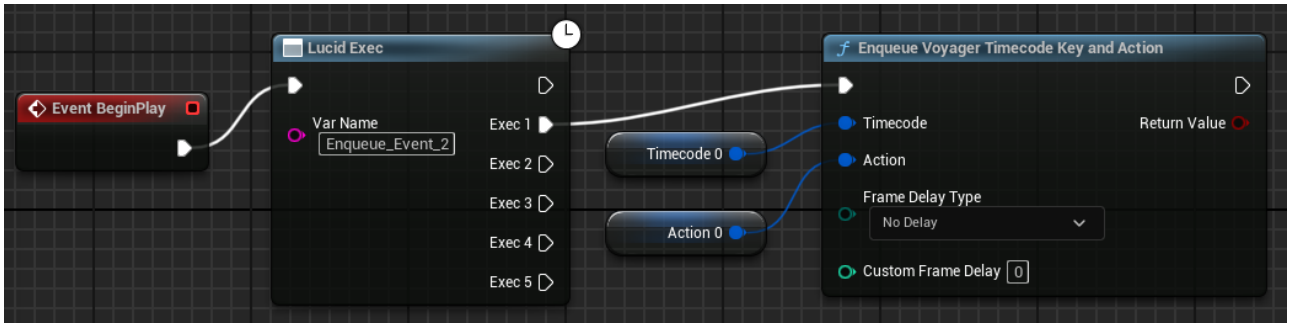
8. Add an input node (e.g., RossTalk GPI or Lucid Exec) and connect the execution pin to the **Exec** input pin of the **Enqueue Voyager Timecode Key and Action** node.

   When Voyager receives the signal from the input node, it will enqueue the event.



Enqueue Voyager Timecode Key and Action - From Lucid Exec Node

9. Add the **Event BeginPlay** node to the input node.



Enqueue Voyager Timecode Key and Action - Add Event Begin Play Node

10. Select **Save** and then **Compile**.

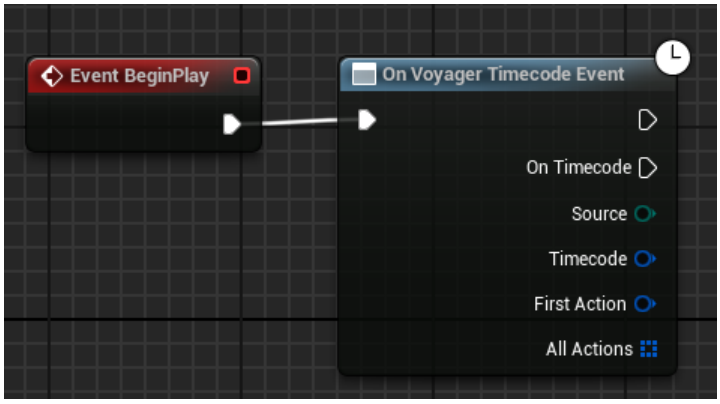
On Voyager Timecode Event

Use the **On Voyager Timecode Event** node to execute an event that has been scheduled using one of the **Voyager Timecode Event** nodes or through **RossTalk**.

See Setting up Timecode Events in the Level Blueprint for instructions on accessing **Voyager - Timecode** nodes.

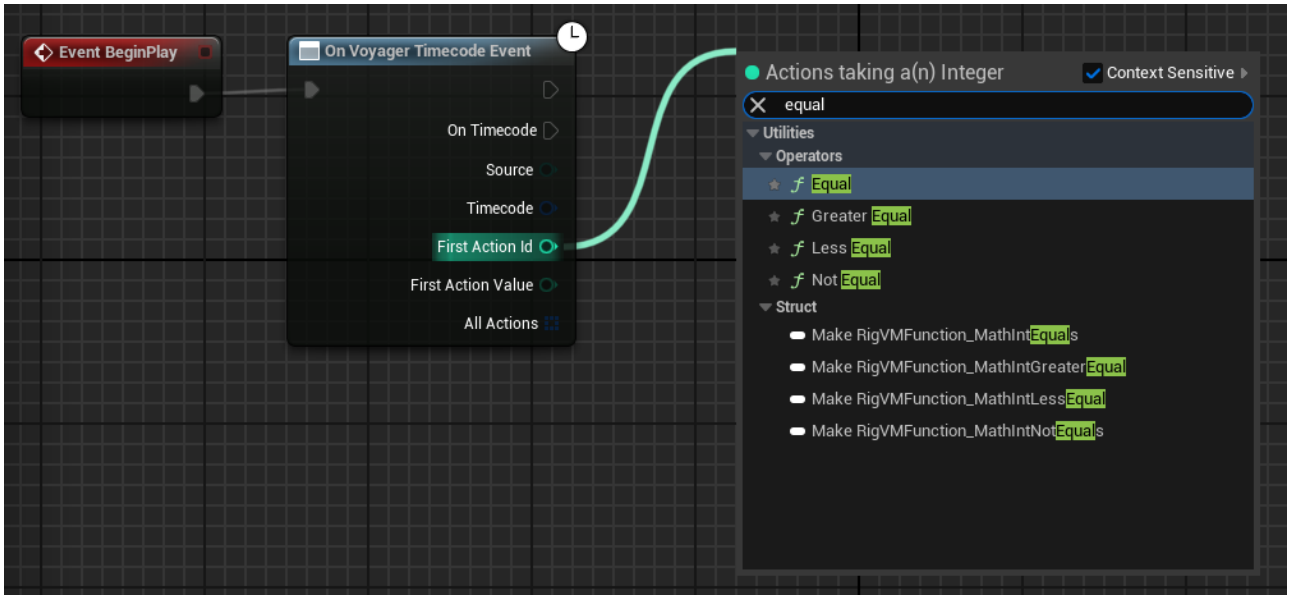
To execute an event with the On Voyager Timecode Event node:

1. Add the **On Voyager Timecode Event** node to the blueprint and then add the **Event BeginPlay** node to initialize it.



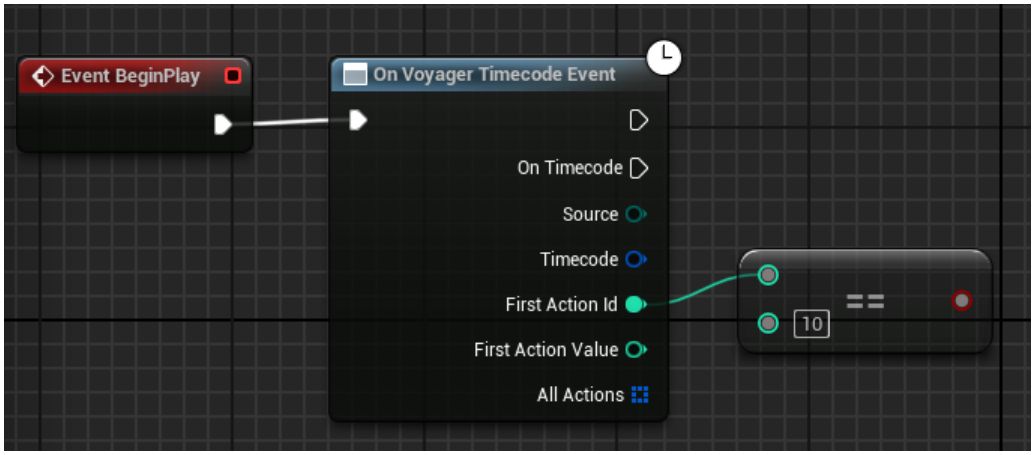
On Voyager Timecode Event

2. Left-click and drag off the **On Timecode** pin and from the **Executable Actions** list, select the event that you want to execute.
3. In the **On Voyager Timecode Event** node, right-click on the **First Action** pin and from the menu, select **Split Struct Pin**.
4. Drag out from the **First Action Id** pin and in the **Actions taking a(n) Integer** list, in the **Search** field, enter `equal` and select the **Equal** operator node.



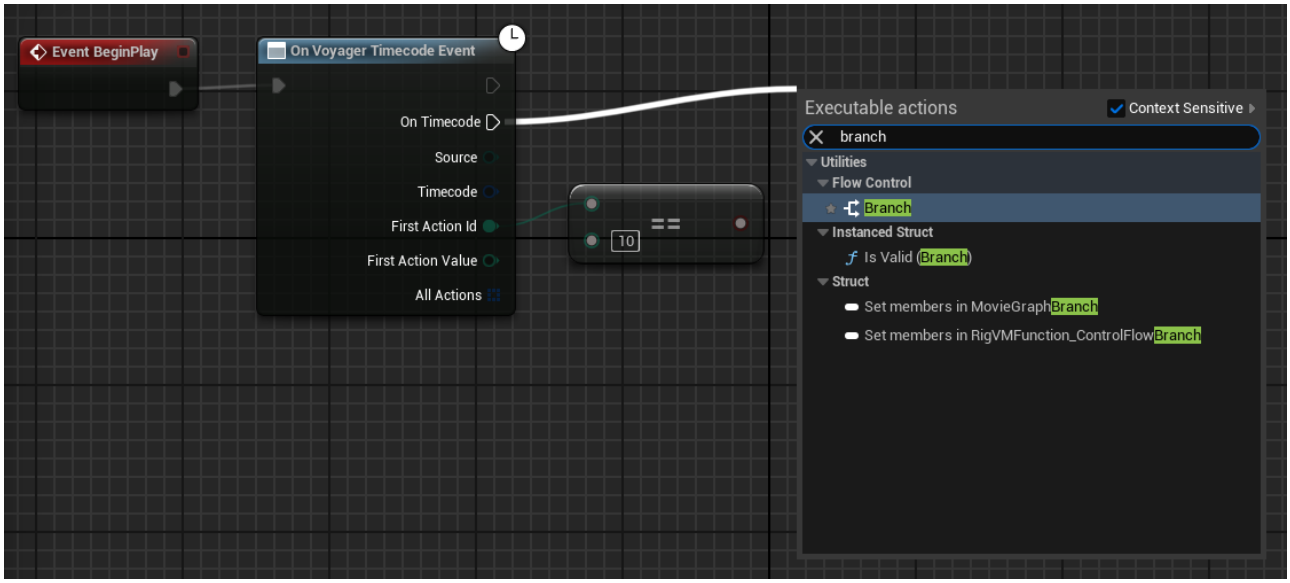
On Voyager Timecode Event - Select Equal Operator

5. In the **Equal** operator node, in the pin field, enter the **Action Id** number of the event you want to trigger.



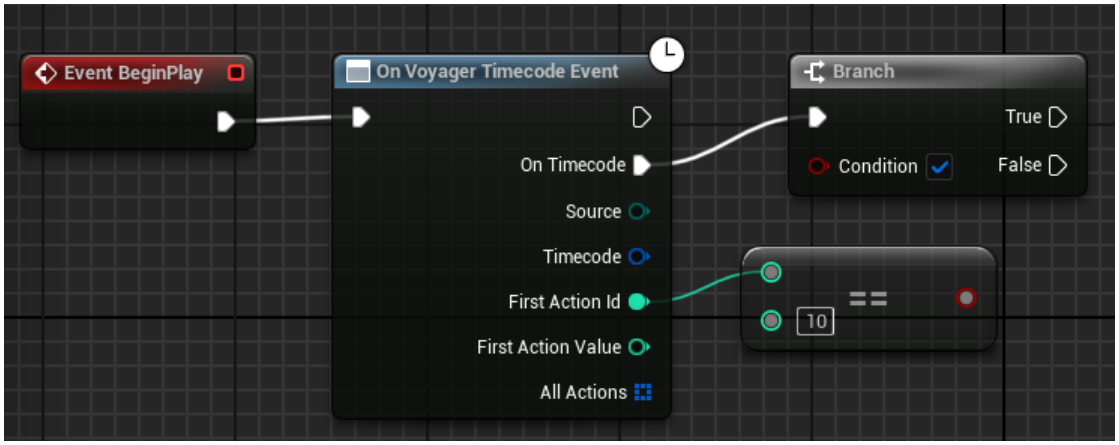
On Voyager Timecode Event - Enter Action Id

6. Drag out from the **On Timecode** pin and in the **Executable Actions** list, in the **Search** field, enter `branch` and select the **Branch** node.



On Voyager Timecode Event - Add Branch Node

The **Branch Node** is added to the blueprint.

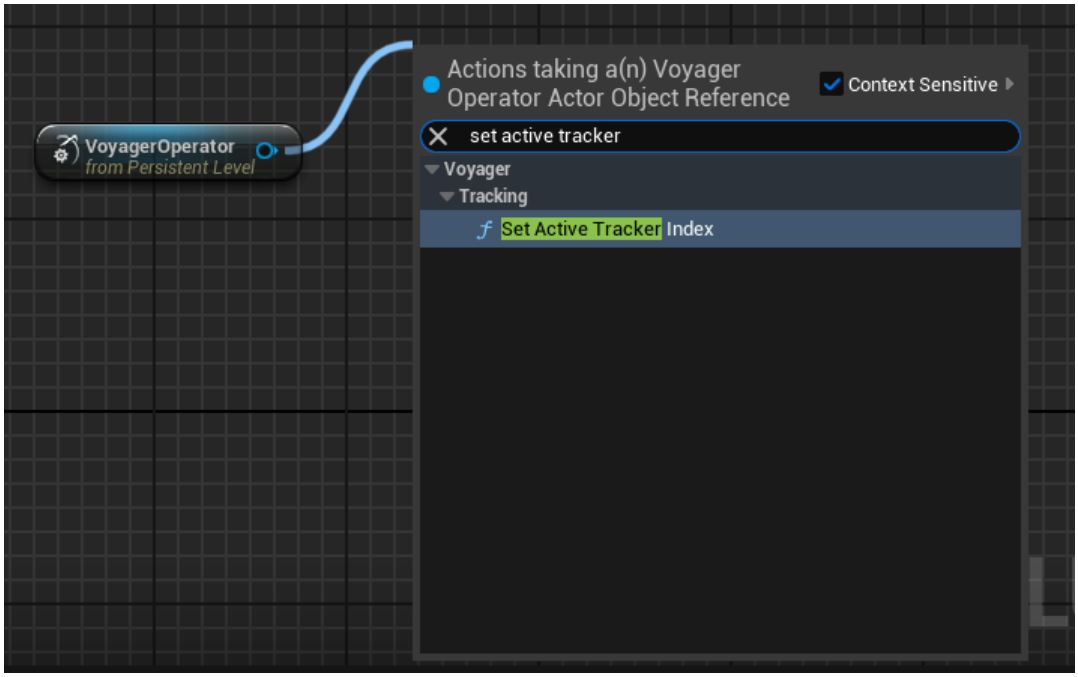


On Voyager Timecode Event - Branch Node Added

7. Continue with the next section to execute a camera switch.

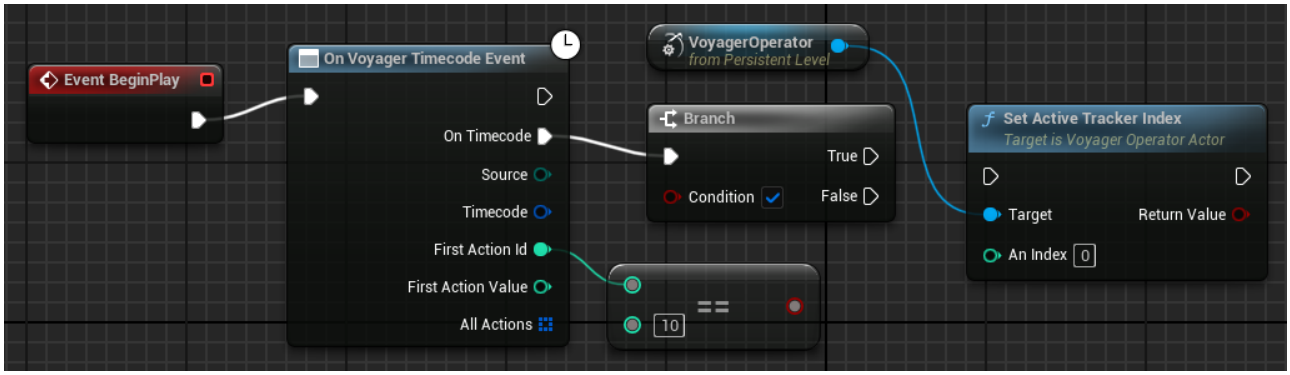
To execute a camera switch using the On Voyager Timecode Event node:

1. After completing the above instructions for executing an event, from the **Outliner**, left-click and drag the **Voyager Operator** actor into the blueprint.
2. Drag out from the **Voyager Operator** node and in the **Actions taking a(n) Voyager Operator Actor Object Reference** list, enter `set active tracker index` and select the **Set Active Tracker Index** node.



On Voyager Timecode Event - Add Set Tracker Index Node

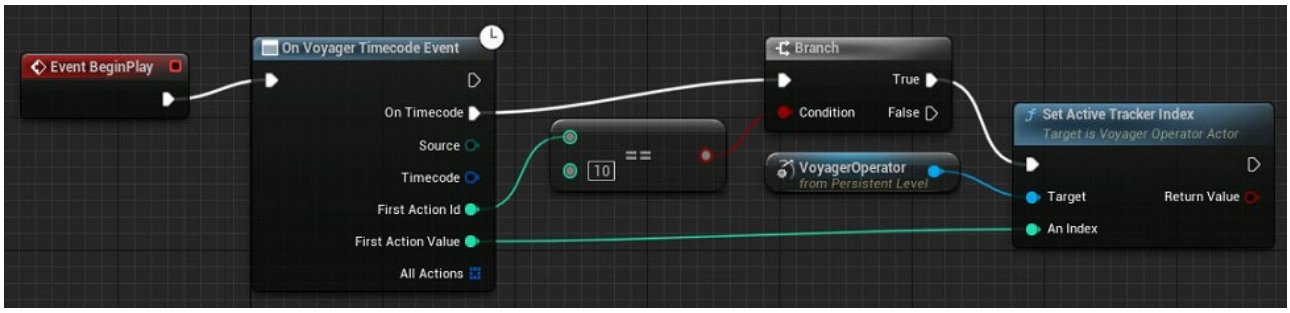
3. The **Set Active Tracker Index** node is added to the blueprint.



On Voyager Timecode Event - Set Active Tracker Index Node Added

4. Make the following connections:

   - Connect the **First Action Value** pin on the **On Voyager Timecode Event** to the **An Index** pin on the **Set Active Tracker Index** node.
   - Connect the **Output** pin on the **Equal Operator** node to the **Condition** pin on the **Branch** node.
   - Connect the **True** output pin on the **Branch** node to the **Exec** pin.



On Voyager Timecode Event - Connect Set Active Tracker Index Node

5. Select **Save** and then **Compile**.

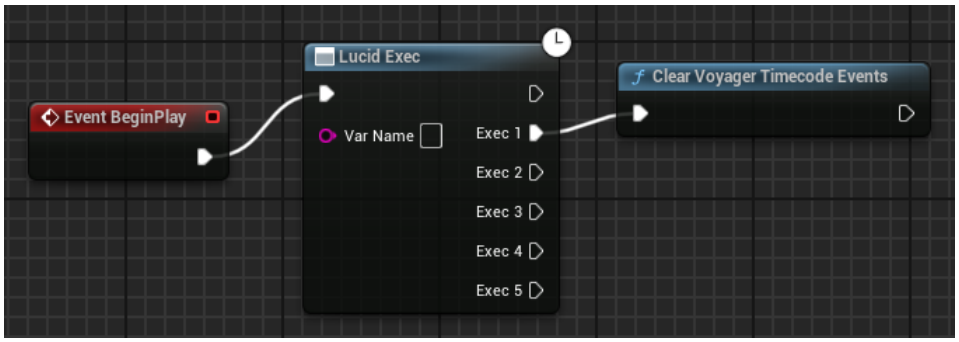
   The **On Voyager Timecode Event** node will now set the active tracker index (the number of the camera) when the **Action Id** used to schedule the event matches the value in the **Equal** operator node.

Clear Voyager Timecode Events

Use this option to clear the list of pending timecode-based events that are scheduled.

To clear Voyager Timecode Events:

1. Add the **Clear Voyager Timecode Events** node to the blueprint.
2. Add an input node (e.g., RossTalk GPI or Lucid Exec) and connect the execution pin to the **Exec** input pin of the **Clear Voyager Timecode Events** node.
3. Add the **Event BeginPlay** node to the input node.



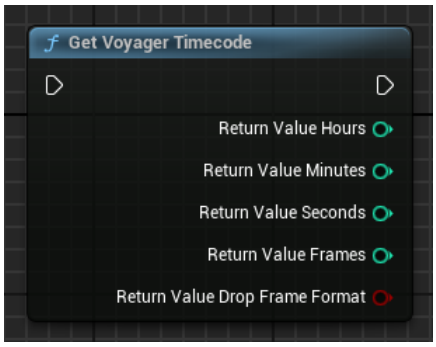
Clear Voyager Timecode Events

Get Voyager Timecode

Use the **Get Voyager Timecode** node to return the current Timecode value of the system.

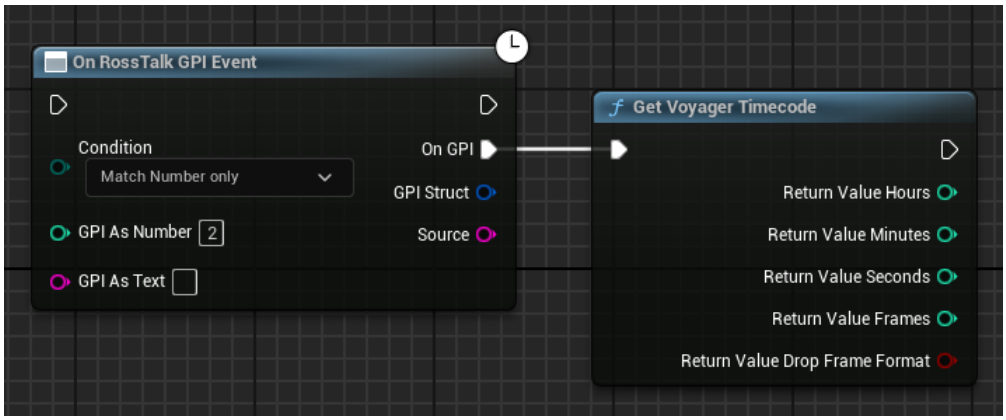
To use the Get Voyager Timecode node:

1. In the **Get Voyager Timecode** node, right-click **Return Value** and select **Split Struct Pin**.



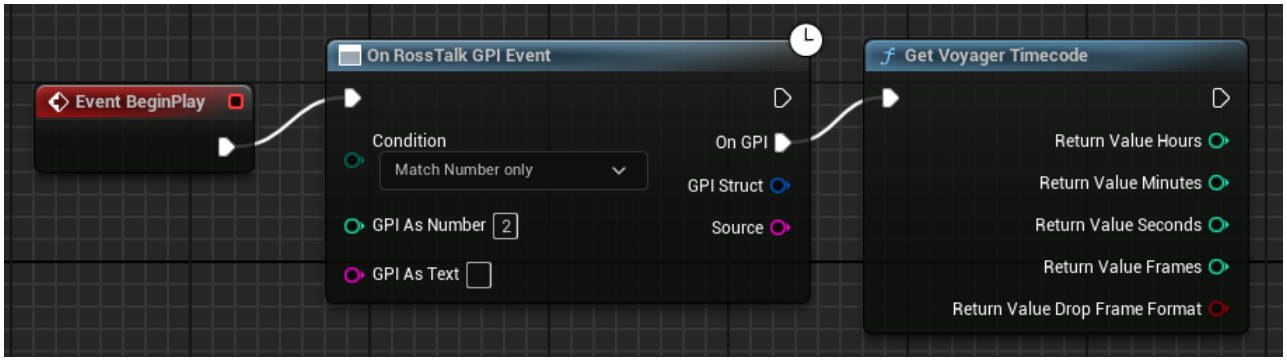
Get Voyager Timecode - Split Struct Pin

2. Add an input node (e.g., RossTalk GPI or Lucid Exec) to trigger the **Get Voyager Timecode** node and connect the execution pin to the **Input** pin of the **Get Voyager Timecode** node.



Get Voyager Timecode - Add Input Node

3. Add the **Event BeginPlay** node to the input node.



Get Voyager Timecode - Add Event Begin Play

4. Drag out from any of the **Return Value** pins of the **Get Voyager Timecode** node and from the **Actions taking a(n) Integer** list, select what you want to do with the returned value.

5. Connect the **Exec** output pin of the **Get Voyager Timecode** node to the **Exec** input pin of the action node selected in Step 4.

6. Select **Save**.

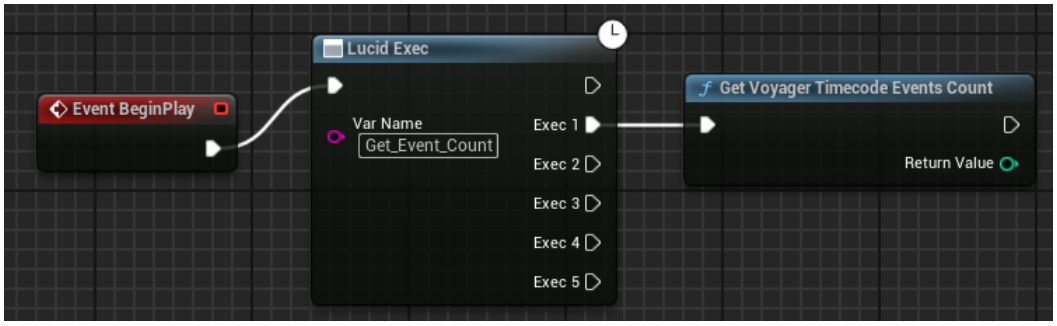
   When the event is executed, the selected return value is retrieved.

Get Voyager Timecode Events Count

Use the **Get Voyager Timecode Events Count** node to return the total number of pending events that are currently scheduled in queue.

To use the Get Voyager Timecode Events Count node:

1. Add an input node (e.g., On RossTalk GPI Event, or Lucid Exec) to the **Get Voyager Timecode Events Count** node.



Get Voyager Timecode Events Count Node

2. Add the **Event BeginPlay** node to the input node.
3. Connect the execution pin of the event node to the **Exec** input pin of the **Get Voyager Timecode Events Count** node.
4. Drag out from the **Return Value** pin of the **Get Voyager Timecode Events Count** node, and from the **Actions taking a(n) Integer** list, select an action node.
5. Connect the **Exec Output** pin of the **Get Voyager Timecode Events Count** node to the **Exec Input** pin of the action node selected in Step 3.
6. Select **Compile** and **Save**.

   If the action event is activated before the events are enqueued, the return value will be zero.

   If the action event is activated after events are enqueued, the number of available events is returned.

Using Voyager Charts

Use Voyager charts to add a graphic representation of data from your DataLinq source. You can choose from pie charts, column/bar charts, and line charts, including multi-line charts.

This feature requires that the Voyager Charts Plugin and the DataLinq Plugin be enabled and that you have a valid DataLinq Plugin License.

For instructions on enabling the Voyager Charts Plugin and DataLinq Plugin, see Enabling the Voyager Plugins.

You will need to have a DataLinq source configured to populate your chart. For more information, see Adding a DataLinq Source to the XPression DataLinq Server.

Pie Charts

Column/Bar Charts

Line Charts

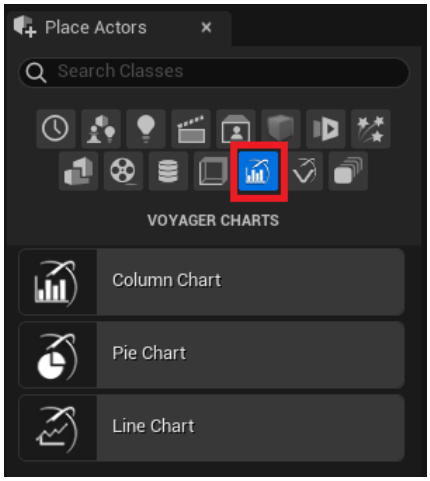
Multi-Line Charts

Creating a Pie Chart

When you create a pie chart, the values, labels and colors come from the source data.

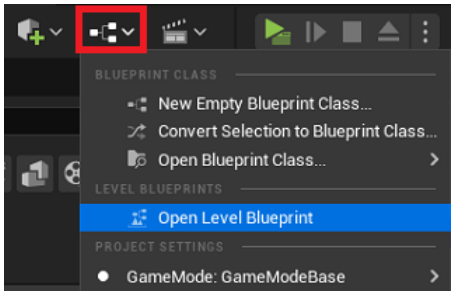
To create a Pie chart:

1. Add a **DataLinq Multi-Value Actor** to the level and configure it as described in Adding a DataLinq Multi Value Actor.
2. In the **Place Actors** tab, from the **Voyager Charts** category, select a **Pie Chart** and drag it into the level.



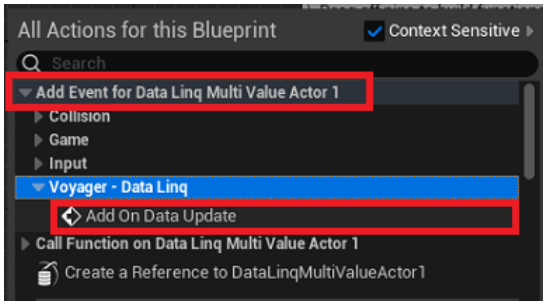
Select a Voyager Chart

3. Select the **Blueprint** icon and select **Open Level Blueprint**.



Open Level Blueprint

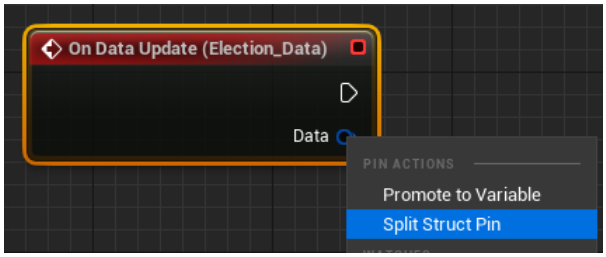
4. Select your **DataLinq Multi-Value Actor** in the **Outliner** and then right-click in an empty area of the blueprint, expand **Add Event for DataLinq MultiValue Actor**, expand **Voyager - DataLinq** and select **Add On Data Update**.



DataLinq Blueprint - Add On Data Update

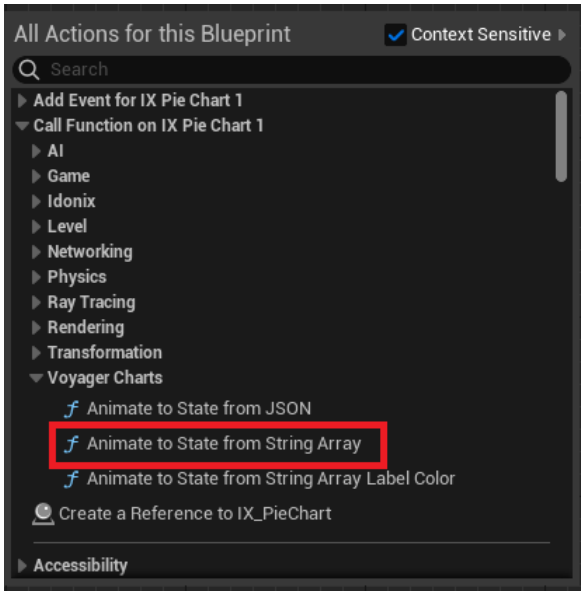
The **On Data Update** node is added to the blueprint.

5. In the **On Data Update** node, right-click the **Data** pin and select **Split Struct Pin**.



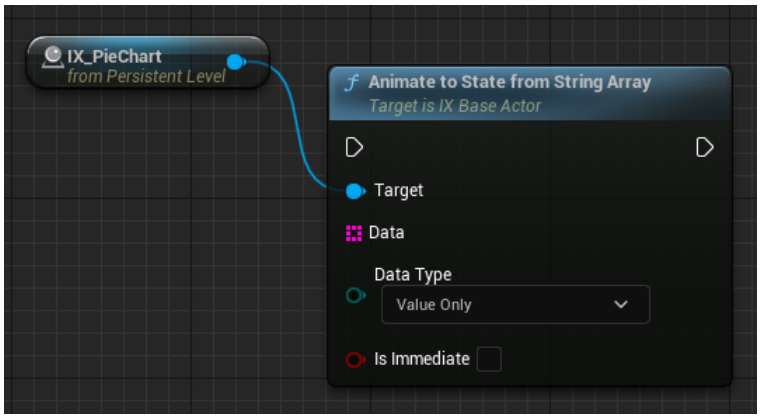
DataLinq Blueprint - Split Struct Pin

6. With your chart selected in the **Outliner**, right-click in an empty area of the blueprint and in the **All Actions for this Blueprint** list, expand **Call Function on...**, expand **Voyager Charts**, and then select **Animate to State from String Array**.



Voyager Charts - Call Function

The reference object for the chart and the **Animate to State from String Array** node are added to the blueprint.



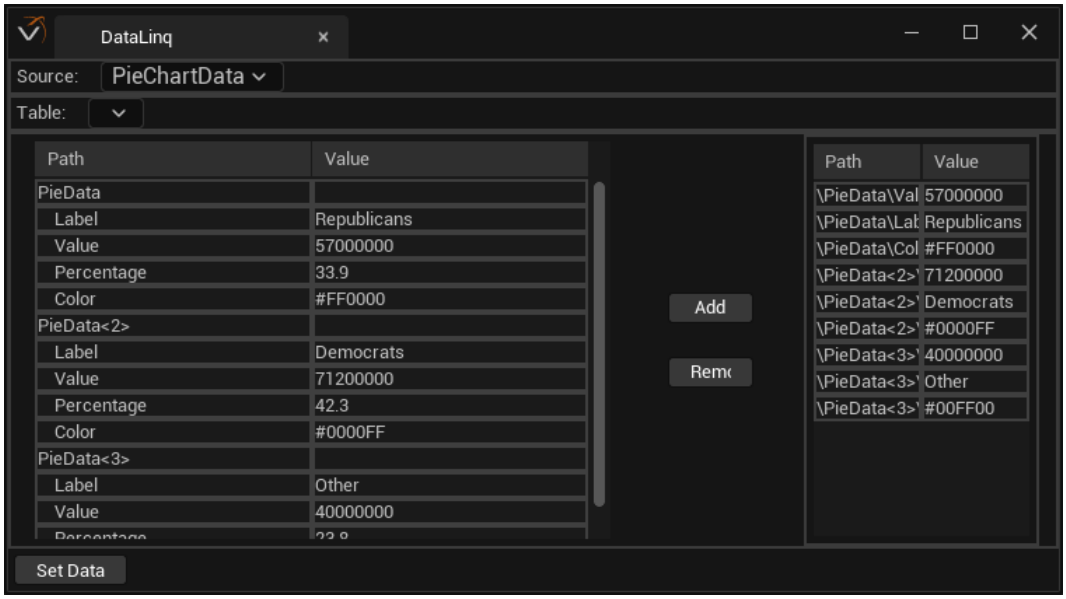
Voyager Charts - Animate to State from String Array

7. In the **Animate to State from String Array** node, from the **Data Type** drop-down, select the data you want to display in the chart.

   The options are:

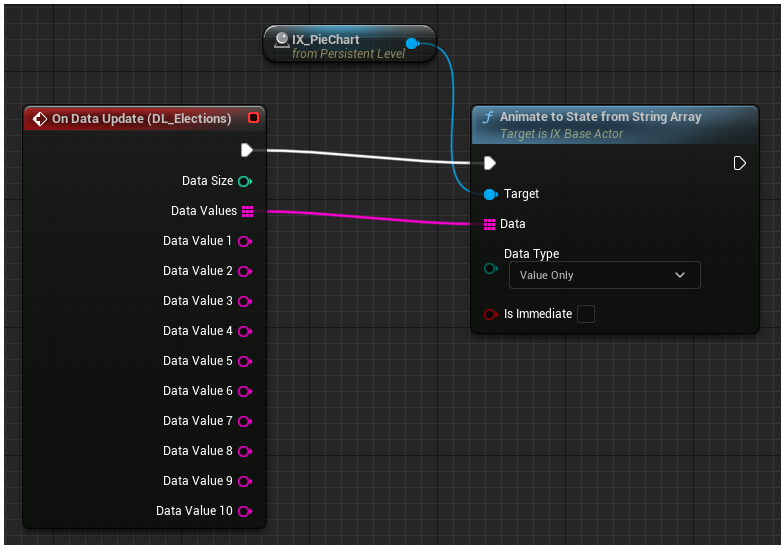
   - **Value Only**
   - **Value and Label** (in your data source, make sure that the data is added in the same order)
   - **Value, Label and Color** (in your data source, make sure that the data is added in the same order)

   For example, in the image below, the data corresponds to **Value**, **Label** and **Color**.



Voyager Charts - Pie Chart Data

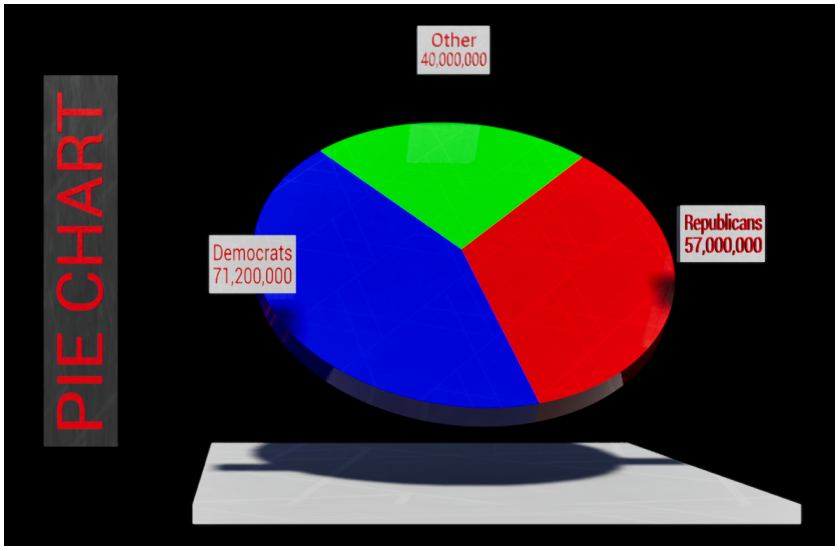
8. Connect the **On Data Update** node as follows:
   - Connect the **Exec** pin of the **On Data Update** node to the **Exec** pin of the **Animate to State from String Array** node.
   - Connect the **Data Values** pin of the **On Data Update** node to the **Data** pin of the **Animate to State from String Array** node.



Voyager Charts - Pie Chart Blueprint

9. Save and compile your blueprint.
10. Press the **Play (PIE)** button to see your chart automatically generated with an IN animation displaying your DataLinq data.

Play Button



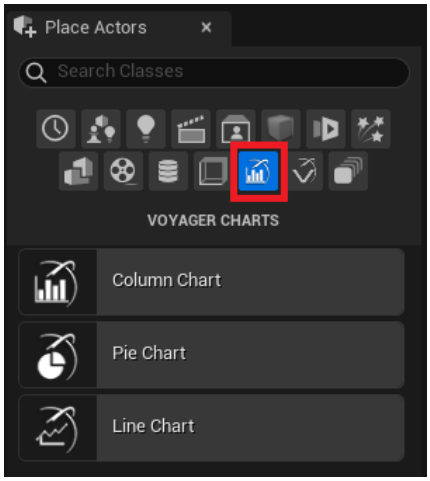
Voyager Charts - Pie Chart Example

Creating a Column/Bar Chart

When you create a column/bar chart, the values, labels and colors are retrieved from the source data.

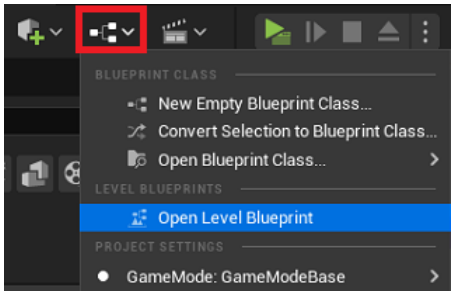
To create a Column/Bar chart:

1. Add a **DataLinq Multi-Value Actor** to the level and configure it as described in Adding a DataLinq Multi Value Actor.
2. In the **Place Actors** tab, from the **Voyager Charts** category, select a **Column Chart** and drag it into the level.



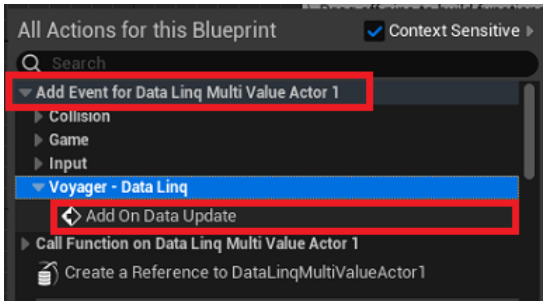
Select a Voyager Chart

3. Select the **Blueprint** icon and select **Open Level Blueprint**.



Open Level Blueprint

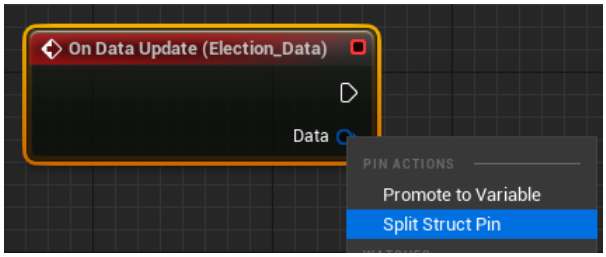
4. Select your **DataLinq Multi-Value Actor** in the **Outliner** and then right-click in an empty area of the blueprint, expand **Add Event for DataLinq MultiValue Actor**, expand **Voyager - DataLinq** and select **Add On Data Update**.



DataLinq Blueprint - Add On Data Update

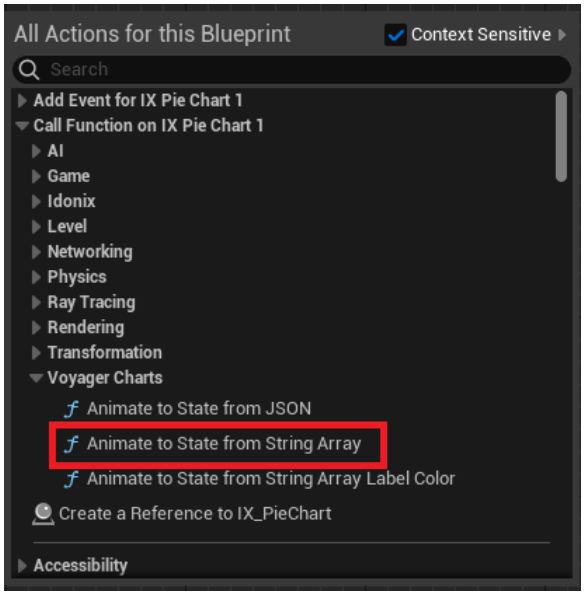
The **On Data Update** node is added to the blueprint.

5. In the **On Data Update** node, right-click the **Data** pin and select **Split Struct Pin**.



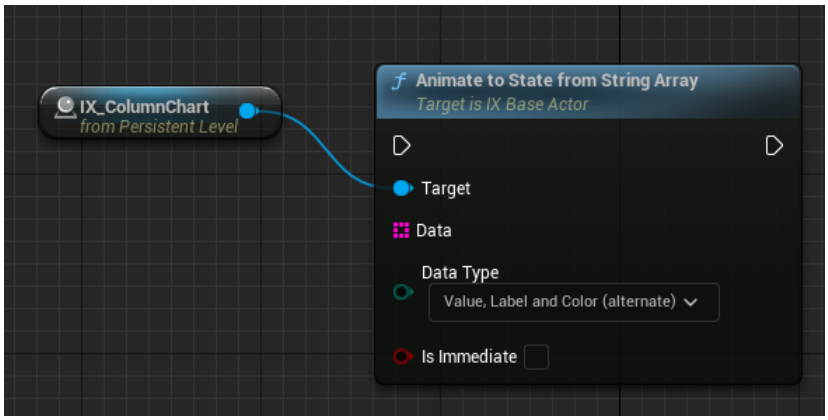
DataLinq Blueprint - Split Struct Pin

6. With your chart selected in the **Outliner**, right-click in an empty area of the blueprint and in the **All Actions for this Blueprint** list, expand **Call Function on...**, expand **Voyager Charts**, and then select **Animate to State from String Array**.



Voyager Charts - Call Function

The reference object for the chart and the **Animate to State from String Array** node are added to the blueprint.



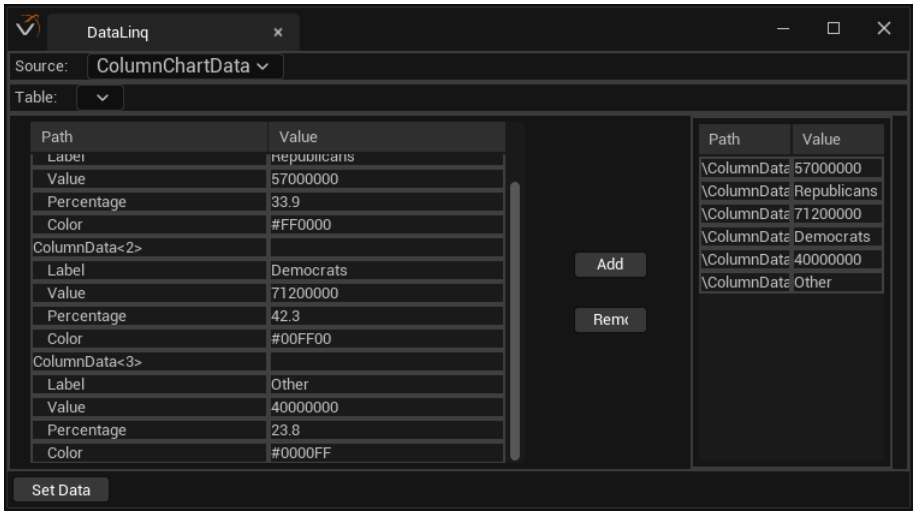
Voyager Charts - Animate to State from String Array

7. In the **Animate to State from String Array** node, from the **Data Type** drop-down, select the data you want to display in the chart.

   The options are:

   - **Value Only**
   - **Value and Label** (in your data source, make sure that the data is added in the same order)
   - **Value, Label and Color** (in your data source, make sure that the data is added in the same order)

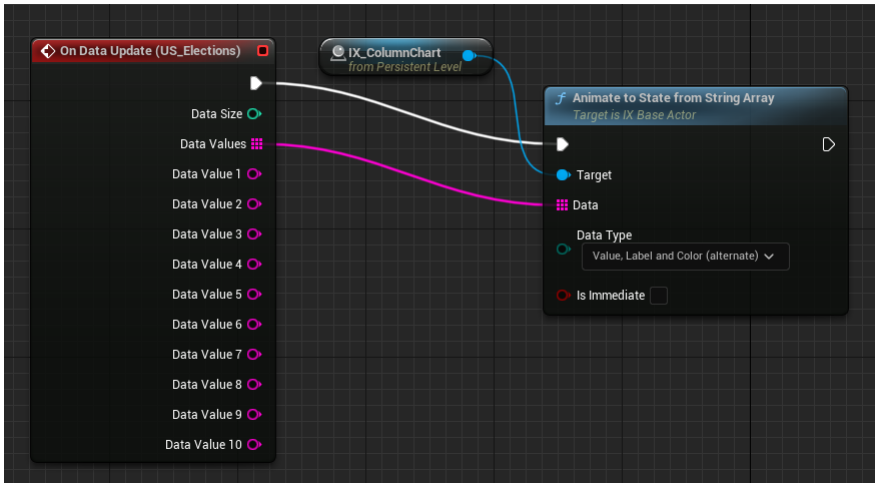
   For example, in the image below, the data corresponds to **Value**, and **Label**.



Voyager Charts - Column Chart Data

8. Connect the **On Data Update** node as follows:

   - Connect the **Exec** pin of the **On Data Update** node to the **Exec** pin of the **Animate to State from String Array** node.
   - Connect the **Data Values** pin of the **On Data Update** node to the **Data** pin of the **Animate to State from String Array** node.



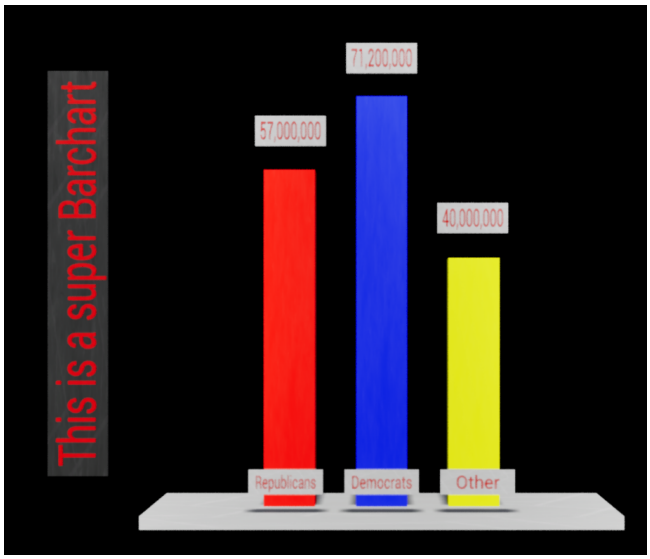
Voyager Charts - Column Chart Blueprint

9. Save and compile your blueprint.

10. Press the **Play (PIE)** button.

Play Button

Your column chart is automatically generated with an IN animation displaying your DataLinq data.



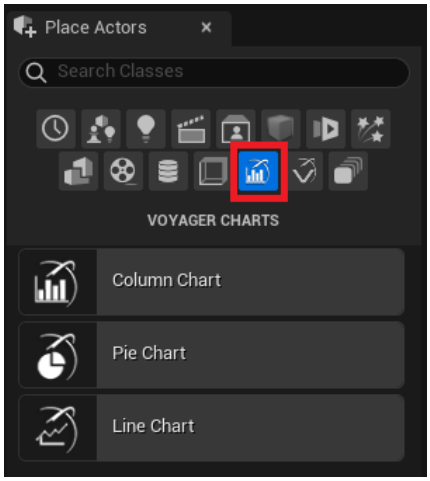
Voyager Charts - Column Chart Example

Creating a Line Chart

When you create a line chart, the values and labels are retrieved from the source data.

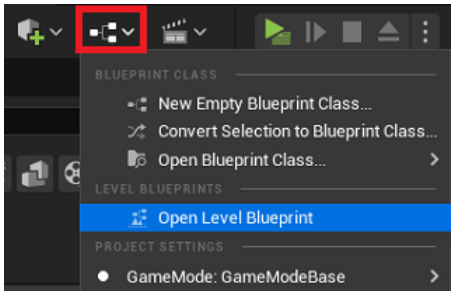
To create a Voyager chart:

1. Add a **DataLinq Multi-Value Actor** to the level and configure it as described in Adding a DataLinq Multi Value Actor.
2. In the **Place Actors** tab, from the **Voyager Charts** category, select a **Line Chart** and drag it into the level.



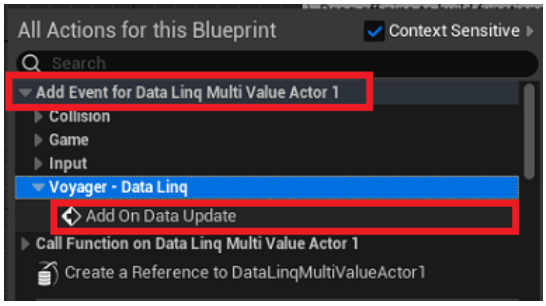
Select a Voyager Chart

3. Select the **Blueprint** icon and select **Open Level Blueprint**.



Open Level Blueprint

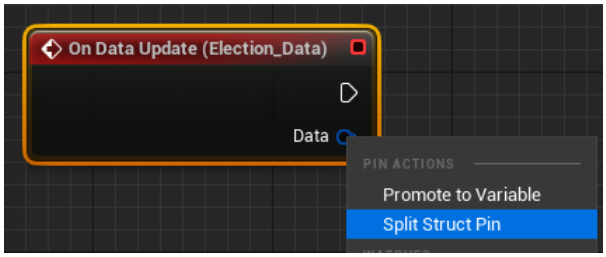
4. Select your **DataLinq Multi-Value Actor** in the **Outliner** and then right-click in an empty area of the blueprint, expand **Add Event for DataLinq MultiValue Actor**, expand **Voyager - DataLinq** and select **Add On Data Update**.



DataLinq Blueprint - Add On Data Update

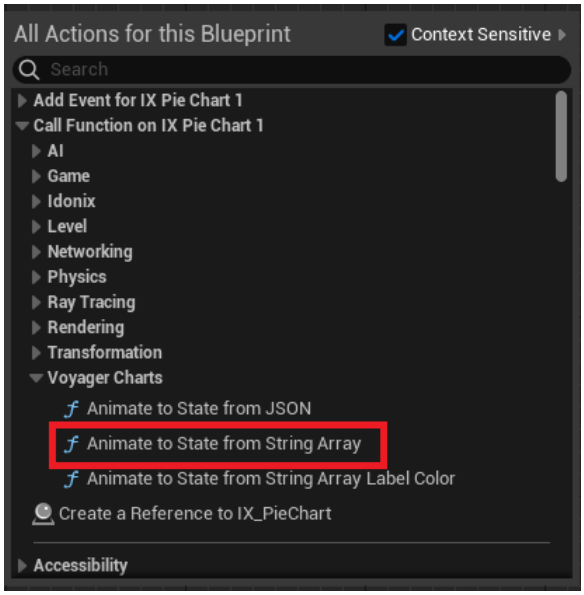
The **On Data Update** node is added to the blueprint.

5. In the **On Data Update** node, right-click the **Data** pin and select **Split Struct Pin**.



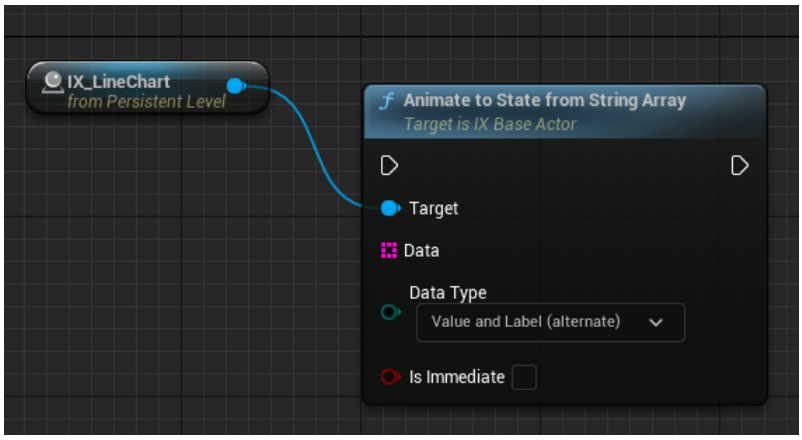
DataLinq Blueprint - Split Struct Pin

6. With your line chart selected in the **Outliner**, right-click in an empty area of the blueprint and in the **All Actions for this Blueprint** list, expand **Call Function on...**, expand **Voyager Charts**, and then select **Animate to State from String Array**.



Voyager Charts - Call Function

The reference object for the chart and the **Animate to State from String Array** node are added to the blueprint.



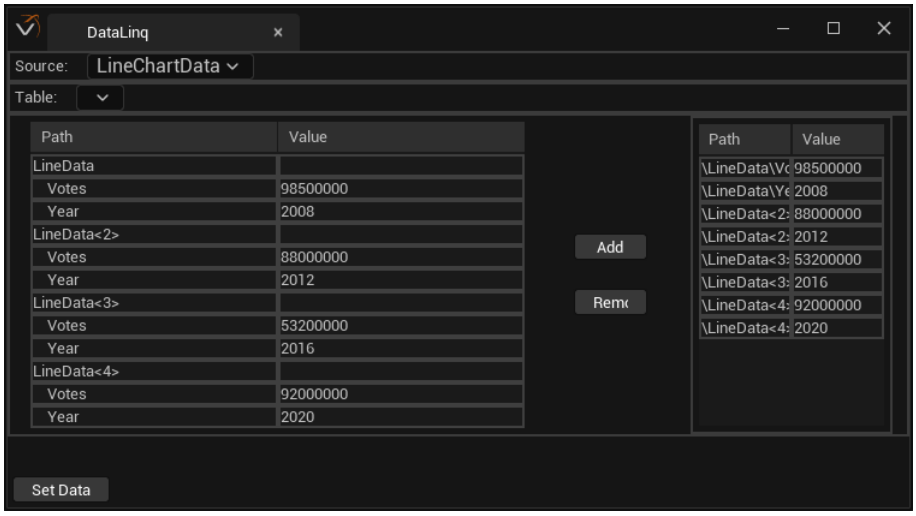
Voyager Charts - Animate to State from String Array

7. In the **Animate to State from String Array** node, from the **Data Type** drop-down, select the data you want to display in the chart.

   The options are:

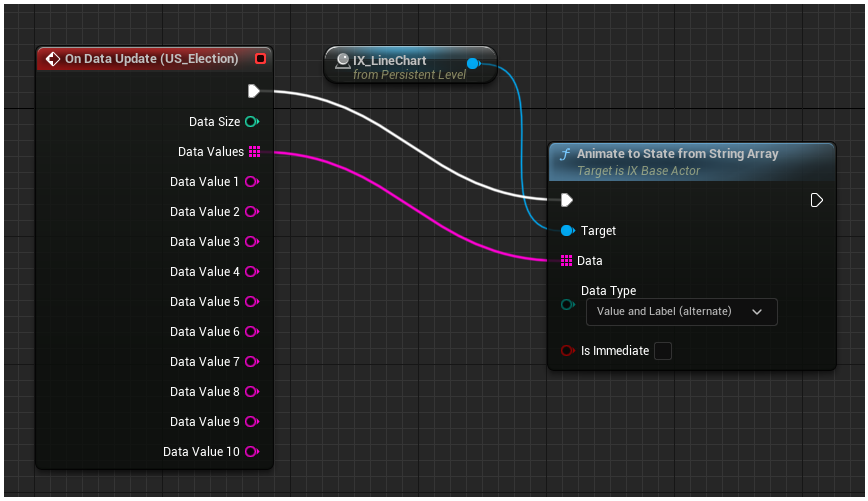
   - **Value Only**
   - **Value and Label** (in your data source, make sure that the data is added in the same order)
   - **Value, Label and Color** (in your data source, make sure that the data is added in the same order)

   For example, in the image below, the data corresponds to **Value** (Votes) and **Label** (Year).



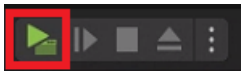
Voyager Charts - Line Chart Data

8. Connect the **On Data Update** node as follows:
   - Connect the **Exec** pin of the **On Data Update** node to the **Exec** pin of the **Animate to State from String Array** node.
   - Connect the **Data Values** pin of the **On Data Update** node to the **Data** pin of the **Animate to State from String Array** node.



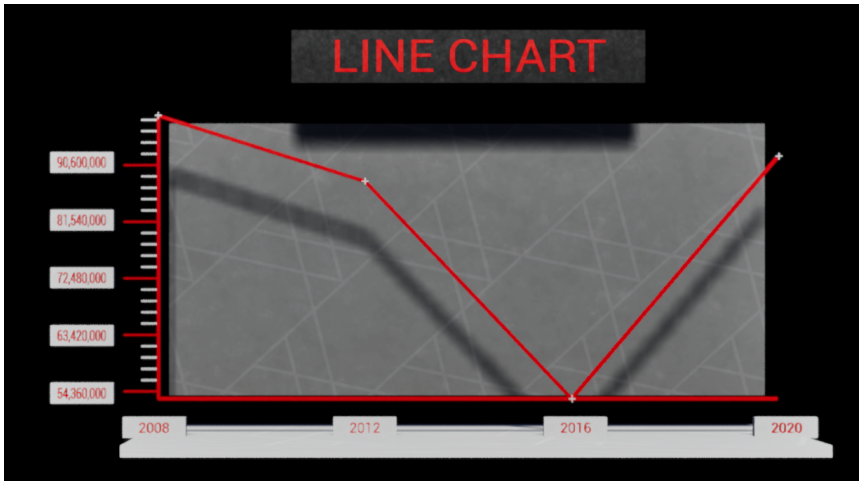
Voyager Charts - Line Chart Blueprint

9. Save and compile your blueprint.
10. Press the **Play (PIE)** button.



Play Button

Your line chart is automatically generated with an IN animation displaying your DataLinq data.



Voyager Charts - Line Chart Example

Creating a Multi-Line Chart

When you create a multi-line line chart, you'll configure the line legends, line colors, and the X Axis labels in the blueprint. You'll also need 2 **DataLinq Multi Value Actors**, one to retrieve the data values for your chart and one to provide the X Axis labels.

Ensure that the number of data values in your DataLinq source is a multiple of the number of lines in your chart. For example, if you have 3 lines then you need to have 3, 6, 9, etc. data values so that each line will have the same number of points.

To create a multi-line chart:

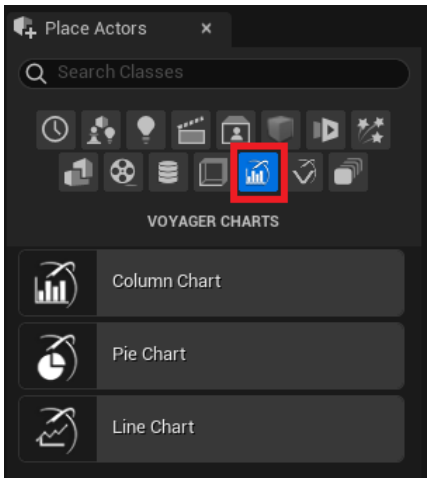
1. Add a **DataLinq Multi Value Actor** to the level and configure it as described in Adding the DataLinq Multi Value Actor.

   This **DataLinq Multi Value Actor** will draw only the values from the data source. All the values for the first line should be selected first, followed by all the values for the second line, etc.

2. Add a second **DataLinq Multi Value Actor** to the level and configure it as described in Adding the DataLinq Multi Value Actor.

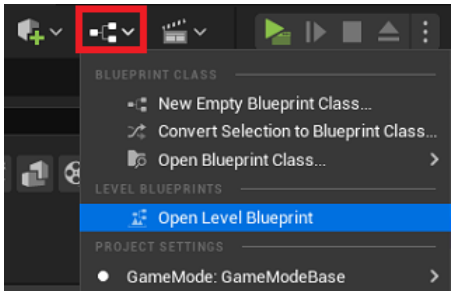
   This **DataLinq Multi Value Actor** will draw only the X Axis labels from the data source.

3. In the **Place Actors** tab, from the **Voyager Charts** category, select the **Line Chart** and drag it into the level.



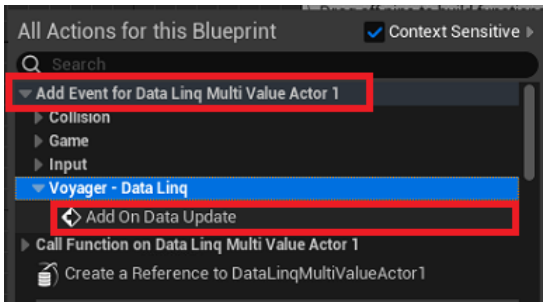
Select a Voyager Chart

4. Select the **Blueprint** icon and select **Open Level Blueprint**.



Open Level Blueprint

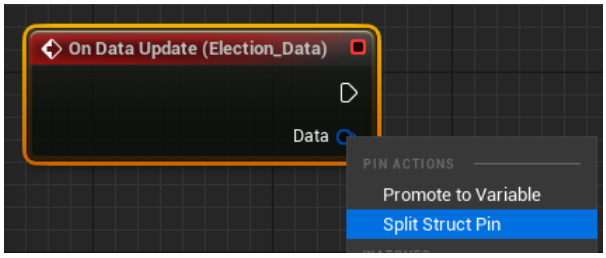
5. In the **Outliner**, select your **DataLinq Multi-Value Actor** and then right-click in an empty area of the blueprint, expand **Add Event for DataLinq MultiValue Actor**, expand **Voyager - DataLinq** and select **Add On Data Update**.



DataLinq Blueprint - Add On Data Update

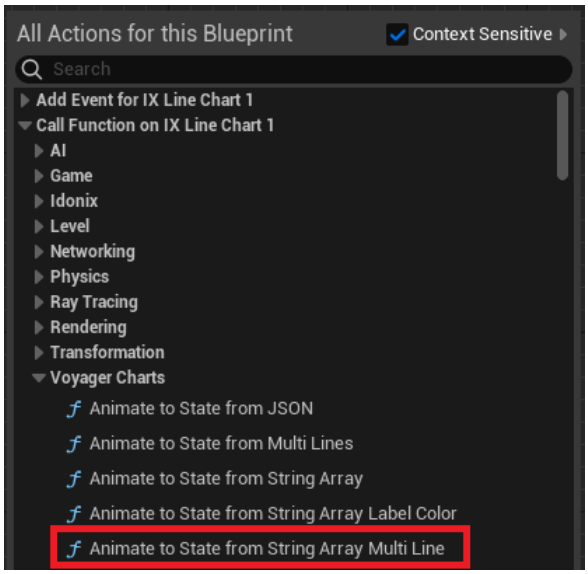
The **On Data Update** node is added to the blueprint.

6. In the **On Data Update** node, right-click the **Data** pin and select **Split Struct Pin**.



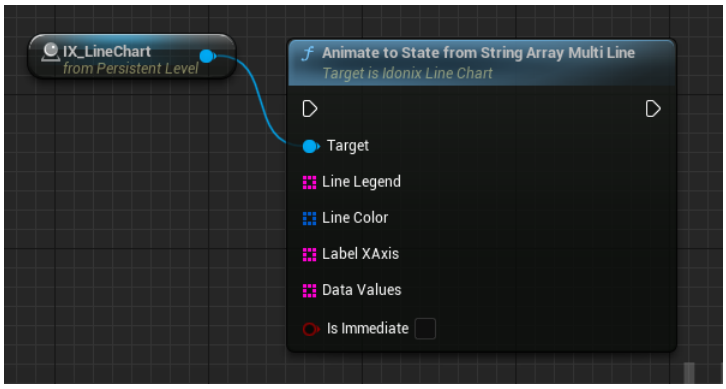
DataLinq Blueprint - Split Struct Pin

7. With your line chart selected in the **Outliner**, right-click in an empty area of the blueprint and in the **All Actions for this Blueprint** list, expand **Call Function on...** , expand **Voyager Charts**, and select **Animate to State from String Array Multi Line**.



Voyager Charts - Add Call Function - Multi Line

The reference object for the chart and the **Animate to State from String Array Multi Line** node are added to the blueprint.



Voyager Charts - Animate to State from String Array Multi Line

To configure the line legends:

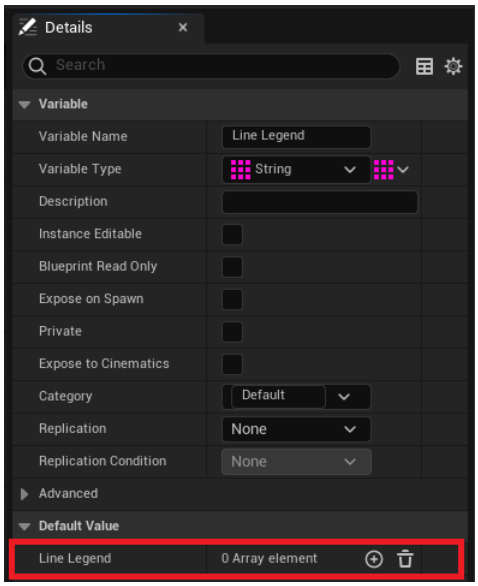
1. In the **Animate to State from String Array Multi Line** node, right-click the **Line Legend** pin and select **Promote to Variable**.

   A **Line Legend** variable node is added to the blueprint.

2. Select **Save** and **Compile**.

3. In the **Variables** list on the left side of the blueprint, select the **Line Legend** variable.

   In the **Details** tab on the right side of the blueprint, under **Default Value**, you'll now see the **Line Legend** variable.



Voyager Charts - Details Tab

4. In the **Details** tab, in the **Line Legend** variable, select the **+** icon beside **Array Elements** to add as many elements as you want lines in your graph.

5. Give each array element a name that corresponds to the data for the line.

To configure the line color:

1. In the **Animate to State from String Array Multi Line** node, right-click the **Line Color** pin and select **Promote to Variable**.

   A **Line Color** variable node is added to the blueprint.

2. Select **Save** and **Compile**.

3. In the **Variables** list on the left side of the blueprint, select the **Line Color** variable.

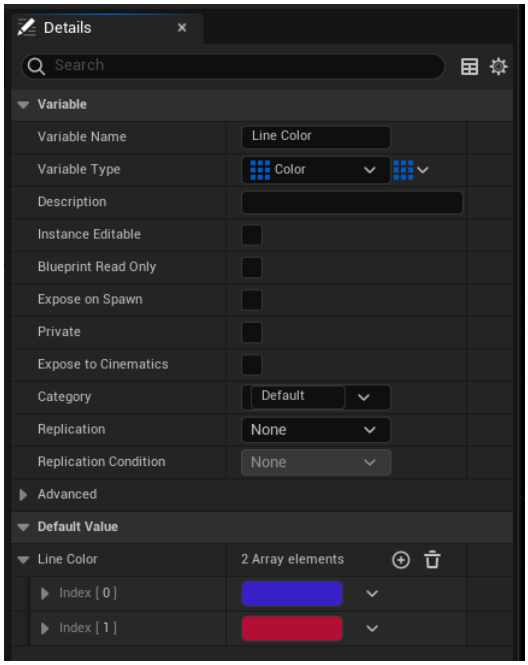
   In the **Details** tab on the right side of the blueprint, under **Default Value**, you'll now see the **Line Color** variable.

4. In the **Details** tab, in the **Line Color** variable, select the **+** icon beside **Array Elements** to add a color for each line in your graph.

5. Double-click the bar for each **Array Element** and use the **Color Picker** to select a color for the line.

   The color for the first line corresponds to the first value in your data source.

   The color for the second line corresponds to the second value in your data source, etc.



Voyager Charts - Details Tab - Line Color

To configure the X Axis labels:

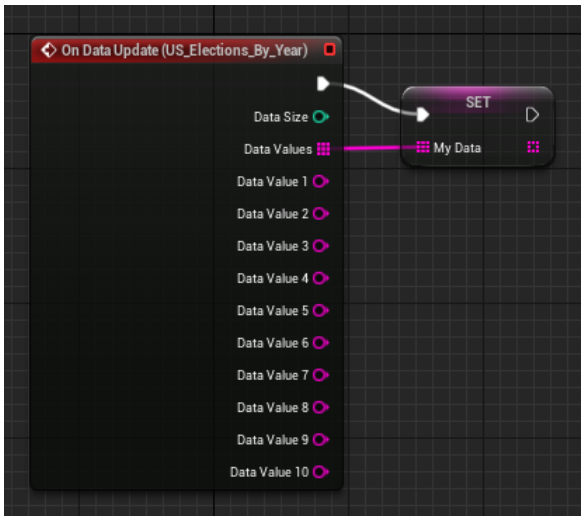
1. In the **Animate to State from String Array Multi Line** node, right-click the **Label X Axis** pin and select **Promote to Variable**.
2. Select **Save** and **Compile**.
3. In the **Variables** list on the left side of the blueprint, select the **Label X Axis** variable.

   In the **Details** tab on the right side of the blueprint, under **Default Value**, you'll now see the **Label X Axis** variable.
4. In the **Details** tab, in the **Label X Axis** variable, select the **+** icon beside **Array Elements** to add an array element for each label on the horizontal axis (X axis) of your chart.
5. In each array element, enter the label name, in the order in which you want them to appear on the chart (which should match the order in which they appear in the data source).

To get values from your data source:

1. In the **Variables** list on the left side of the blueprint, select the **+** icon beside **Variables** to add a new variable to represent the data from your data source and give the variable a name (e.g., My Data).
2. With the new variable selected, in the **Details** tab, from the **Variable Type** drop-downs, select **String** and **Array**.
3. Drag the variable into the blueprint and from the menu, select **Set My Data**.

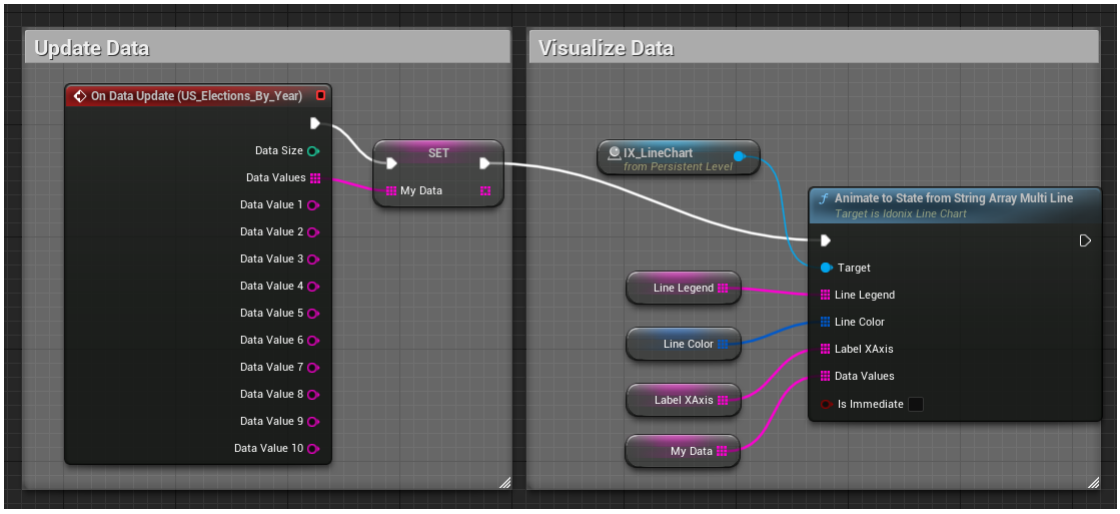
   The **Set** variable node is added to the blueprint.
4. Connect the **On Data Update** node as follows:
   - Connect the **Exec** pin of the **On Data Update** node to the **Exec** pin of the **Set** variable node.
   - Connect the **Data Values** pin of the **On Data Update** node to the **Data Values** pin of the **Set** variable node (the **Data Values** pin will have the name you entered for the **Set** variable).



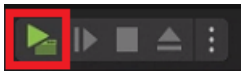
Voyager Charts - Set Data

5. In the **Animate to State from String Array Multi Line** node, left-click and drag off the **Data Values** node and in the **Search** field, begin typing the name of the **Set** variable you created.
6. Then select the **Get** variable node for your **Set** data variable.

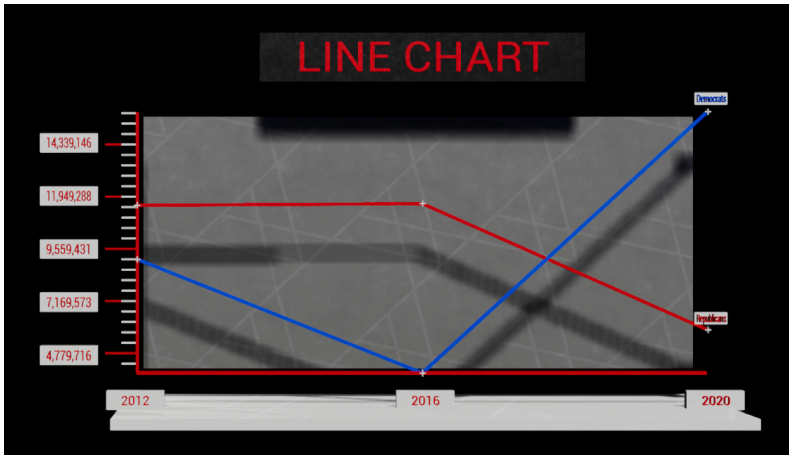
7. Connect the **Output** pin of the **Set Data** node to the **Input** pin of the **Animate to State from String Array Multi Line** node.



8. Save and compile your blueprint.
9. Press the **Play (PIE)** button.



Play Button



Voyager Charts - Multi-Line Chart Example

Timecode

Use the **Timecode** feature to schedule and execute events through Blueprints or using RossTalk. Events can be scheduled to occur at a specific hour, minute, second, and frame.

You will first need to configure the Timecode Provider. Then set up Timecode events using blueprint nodes in the level blueprint or schedule Timecode events using a RossTalk command.

You can also configure frame delays to fine tune the timing of the events.

Configuring the Timecode Provider

The **Timecode Provider** allows the timecode to be read every frame and enables you to schedule and trigger events on certain timecodes. To use the **Timecode Provider**, you will need to configure it inside the media profile.

Once you have configured the **Timecode Provider**, you can then use the Timecode for setting up events in the level blueprint or scheduling them with a RossTalk command.

The following procedures are covered in this section:

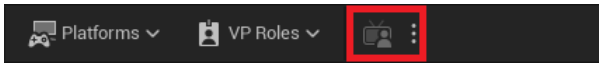
To configure the Timecode Provider - Matrox

To configure the Timecode Provider - Adrienne LTC

To view the Timecode Provider

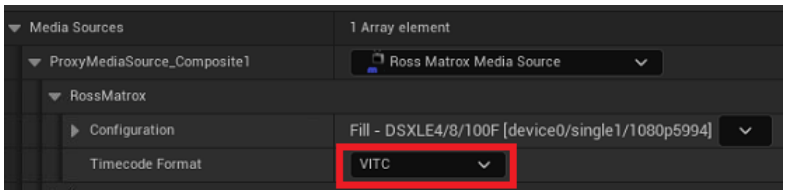
To configure the Timecode Provider - Matrox:

1. Double-click the **Media Profile** icon in the main tool bar to open the editor.



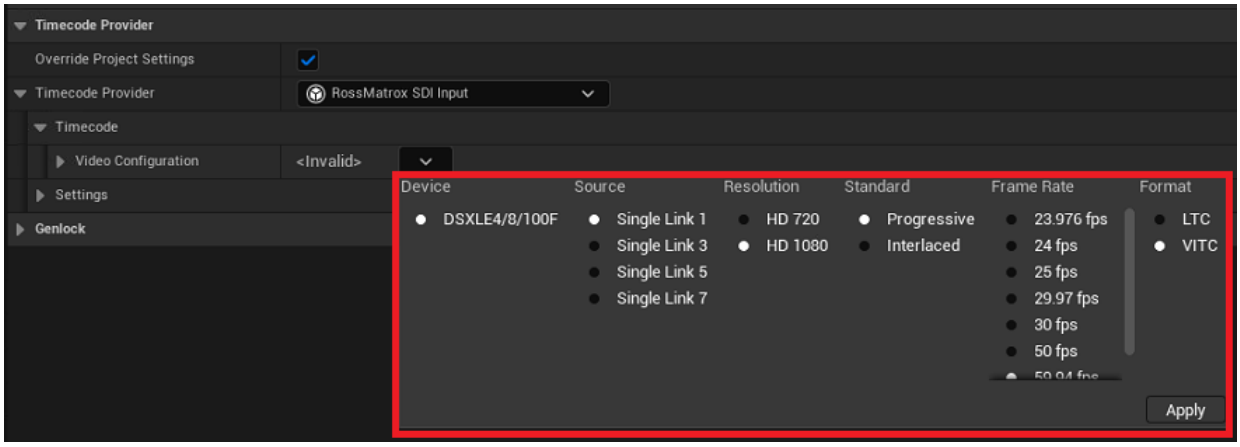
Media Profile Icon

2. In the **Media Profile** editor, expand the proxy media source for the input whose timecode you want to read.
3. Expand **Ross Matrox** and from the **Timecode Format field** drop-down, select **VITC**.



Select Timecode Format

4. Expand the **Timecode Provider** section and select the **Override Project Settings** checkbox.
5. From the **Timecode Provider** drop-down, select **RossMatrox SDI Input**.



Timecode Provider Configuration

6. Expand **Timecode** and select the arrow beside the **Video Configuration** field.

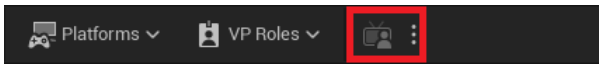
7. Configure the settings to match the input (composite/live source/AR Background) you want to read, selecting **VITC** in the **Format** column, and then select **Apply**.
8. Select **Save** and close the **Media Profile** editor.

To configure the Timecode Provider - Adrienne LTC:

1. Ensure the **Ross Adrienne GPIO Plugin** is enabled.

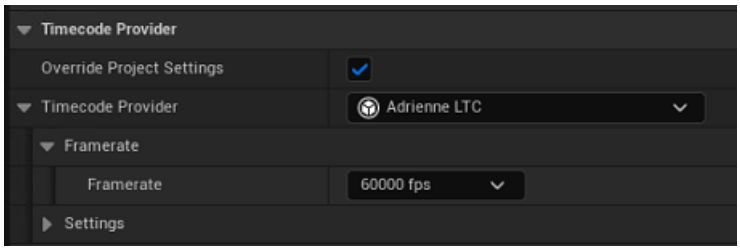
   See Enabling the Voyager Plugins for instructions.

2. Double-click the **Media Profile** icon in the main tool bar to open the editor.



Media Profile Icon

3. Expand the **Timecode Provider** section and select the **Override Project Settings** checkbox.
4. From the **Timecode Provider** drop-down, select **Adrienne LTC**.



Media Profile - Timecode Provider Settings

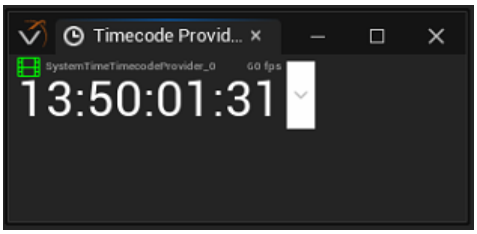
5. Expand **Framerate** and from the **Framerate** drop-down, select the framerate that corresponds to your input signal.
6. Select **Save** and close the **Media Profile** editor.

   The changes have been saved and the project now supports the **Timecode**.

   For information on using the **Timecode** for event scheduling, see the Timecode - Scheduling Events section.

To view the Timecode Provider:

- In the main toolbar, select **Window > Virtual Production > Timecode Provider**.



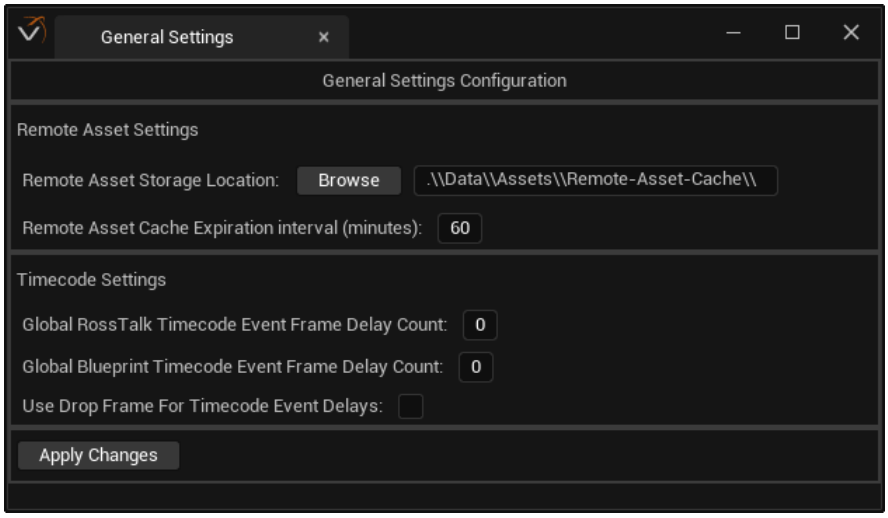
Timecode Provider

Configuring Frame Delays

You can configure a frame delay for timecode events triggered through RossTalk or through a blueprint.

To configure a frame delay:

1. Select **Window > Voyager > General Settings**.



General Settings

2. In the **Timecode Settings** section, enter the number of frames to delay, depending on your event scheduling method and framerate:
   - **Global RossTalk Timecode Event Frame Delay Count** — the delay is added to all scheduled RossTalk Timecode events.
   - **Global Blueprint Timecode Event Frame Delay Count** — the delay is added to all scheduled blueprint Timecode events.

     ★This setting can be overwritten if **No Delay** is selected when scheduling a Timecode event in a blueprint node or if a **Custom Frame Delay** has been set in the node.
   - **Use Drop Frame for Timecode Event Delays** — the delay is added when a fractional framerate (e.g., 29.97 fps) is used, to preserve the overall time accuracy.
3. Then select **Apply Changes** and close the **General Settings** window.

Scheduling Timecode Events Using a RossTalk Command

Timecode events can be scheduled using a custom RossTalk command.

To receive this command, the RossTalk Plugin needs to be enabled. In Voyager, the RossTalk Plugin is enabled by default.

The syntax for the command is as follows:

TC hh:mm:ss:ff aa:vv

Parameter	Description
hh	Hours
mm	Minutes
ss	Seconds
ff	Frame Number
aa	Custom Action ID
bb	Custom Action VA

You can configure a frame delay for timecode events triggered through RossTalk. See Configuring Frame Delays for further information.

Template Links

Template links allow you to manage which Voyager actors, materials, sequences and levels can be accessed and modified in Lucid Studio or Voyager Trackless Studio. Only those items which are published will be available to the other applications. You can also designate whether or not actors are moveable, can be used as media targets, or can have their functions published and specify to which part of an actor you can apply media.

If there are items in your project that already have tags assigned to them from a previous version, you can add the items along with their tags.

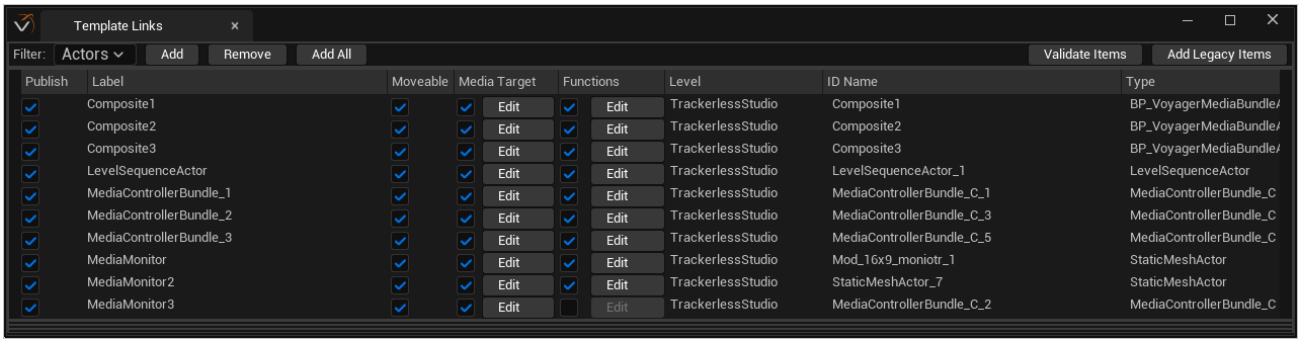
The items are ordered alphabetically and grouped by level and identified by their **ID Name** and **Type**.

You can check that all the items in the list still exist in the project using the Validate Items button.

To open the Template Links window:

- In Voyager, select **Window > Voyager > Template Links**.

  The **Template Links** manager opens.



Template Links Manager

See the following topics for more information:

Adding and Removing Items

Validating Items

Tagging Items

Targeting Selected Material Textures

Adding and Removing Items

Use the buttons along the top of the **Template Links Manager** to add and remove items.

To add items to the Template Links manager:

1. From the **Filter** drop-down, select the type of item to add.

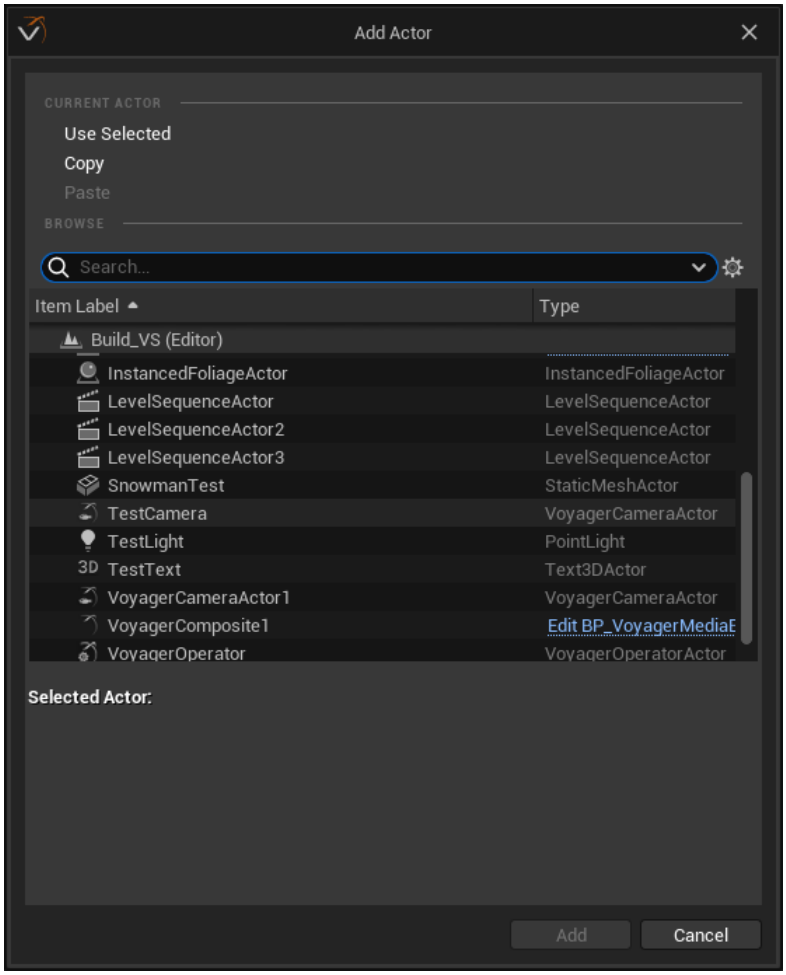
   The options are:

   - **Actors**
   - **Materials**
   - **Sequences**
   - **Levels**

2. Then select **Add**.

   Alternatively, you can right-click anywhere in the list and from the context menu, select **Add Item**.

   The **Add Actor** (or **Add Material** or **Add Sequences** or **Add Levels**) window opens, depending on which item type you selected in the first step.



Template Links - Add Item

3. Scroll through the list to find the item you want to add.

   Alternatively, in the **Search** field, enter the name of the item you want to add.

4. Select the item in the list and then select **Add**.

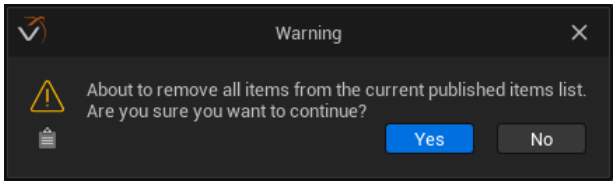
   The selected item is added to the list (in alphabetical order) and automatically tagged to be published.

To remove an item(s):

1. From the **Filter** drop-down, select the type of item to remove.
2. Then select an item in the list and select **Remove**.

   Alternatively, you can right-click an item in the list and from the context menu, select **Remove Item** or right-click in the list and select **Remove All**.

   If you select **Remove All**, a confirmation dialog opens.



Template Links - Remove All Confirmation Dialog

3. Select **Yes** to continue or **No** if you change your mind.

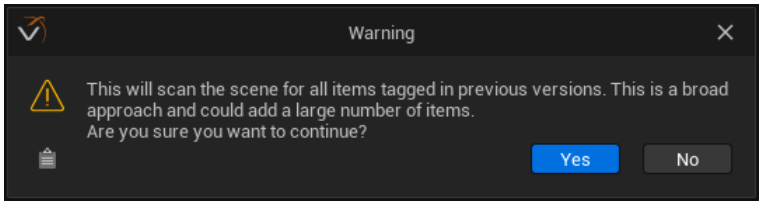
   The selected item or items are removed from the list.

To add Legacy items:

1. From the **Filter** drop-down, select **Actors**.
2. Then select **Add Legacy Items**.

   The project will be scanned and any items that have valid Voyager metadata will be found.

   A confirmation dialog opens, warning that this approach may add a large number of items.



Template Links - Add Legacy Items Confirmation Dialog

3. Select **Yes** to continue or **No** if you change your mind.

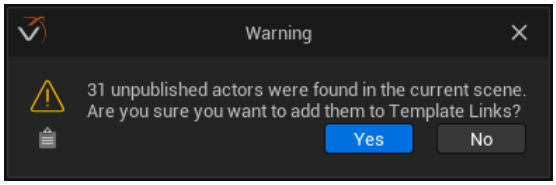
   The legacy items are added to the list with the existing tags selected.

To add all items of the Actor or Sequence type:

1. From the **Filter** drop-down, select **Actors** or **Sequences**.
2. Then select **Add All**.

   Alternatively, you can right-click anywhere in the list and from the context menu, select **Add All**.

   A confirmation dialog opens which tells you how many items of that type were found. This could be a very large number of items, depending on the project. Be sure you want to add all of them before confirming.



Template Links - Add All Confirmation Dialog

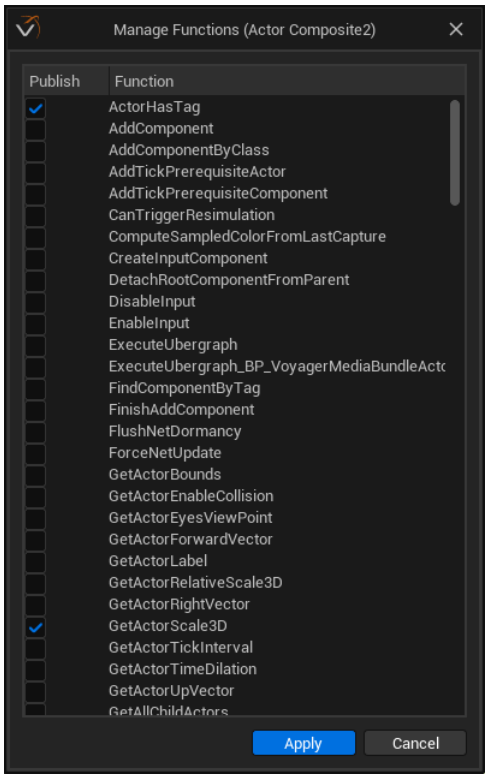
3. Then select **Yes** to add all the items or **No** if you change your mind.

   Items in the project of that type are added to the list (in alphabetical order) and automatically tagged to be published.

To select functions to be published:

1. For an **Actor** or **Level** item, select the **Edit** button in the **Functions** column.

   The **Manage Function** panel opens.



Template Links - Manage Functions Panel

2. Select the functions that you want to publish.
3. Then select **Apply**.

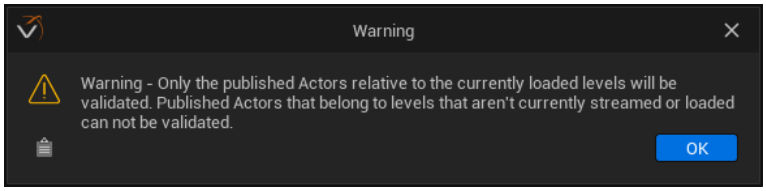
Validating Items

You can check that all the items in the Template Links Manager still exist in the project and remove any that don't.

To validate items:

1. Select the **Validate Items** tab.

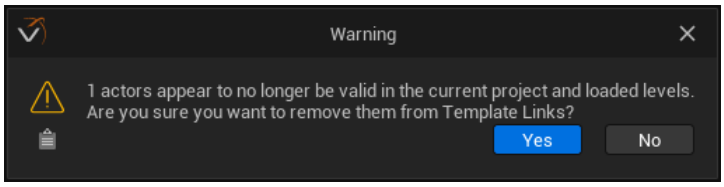
   A confirmation dialog opens indicating that only those items in the currently loaded level will be validated.



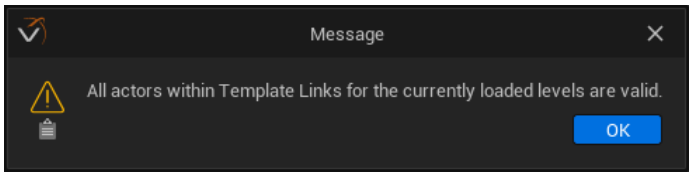
Template Links - Validation Limitation Dialog

2. Select **Yes** to continue.

   A confirmation dialog opens indicating how many items are no longer valid (they do not exist in the project), or indicating that all the items are valid (they still exist in the project).



Template Links - Invalid Items Dialog



Template Links - All Items Valid Dialog

3. Select **Yes** to remove any invalid items from the list.

Tagging Items

You can tag individual items in the **Template Links Manager** to control which items are used by other applications and how they can be used.

To tag items:

1. From the **Publish Type Filter** drop-down, select the type of item to tag.
2. In the list, do any of the following:
   - Select the checkbox in the **Publish** column to make it available to other applications or clear the checkbox to just keep a list of items in the project, but not have them available to other applications.
   - Select the checkbox in the **Moveable** column to make it possible to change the item's position from another application (applies to the **Actor** type item only).
   - Select the checkbox in the **Media Target** column to make it possible to add movies or stills to the item from Lucid Studio or Voyager Trackless (applies to the **Actor** type item only).
   - Select the **Edit** button in the **Media Target** column to open the **Manage Media Targets** window, where you can select to which texture materials you want to allow media to be applied.
   - Select the checkbox in the **Functions** column to make it possible to call the functions associated with that item from Lucid Studio (applies to the **Actor** and **Level** type item only).
   - Select the **Edit** button in the **Functions** column to open the **Manage Functions** window, where you can select which of the actor's (or level's) functions you want to publish.

   ★You can select multiple items at once and select a tag to tag them all.

Targeting Selected Material Textures

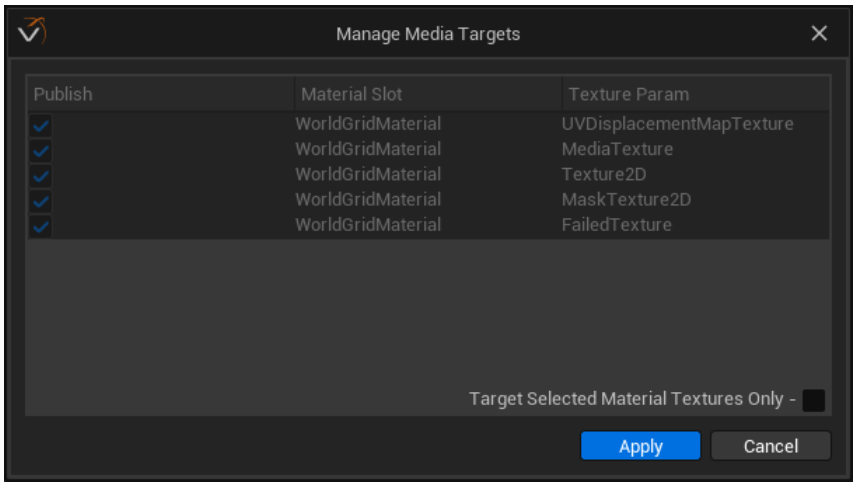
Each actor can have multiple material textures, for example a video monitor could have material textures for the frame and for the screen itself. You can select only those material textures that you want to be able to apply media to, in the **Manage Media Targets** window.

It is helpful to give meaningful names to material slots in the actor mesh. See Naming Material Slots for more information.

To specify to which material texture(s) media can be applied:

1. In the **Media Target** column, select the **Edit** button for the target you want to edit.

   The **Manage Media Targets** window opens.



   Manage Media Targets

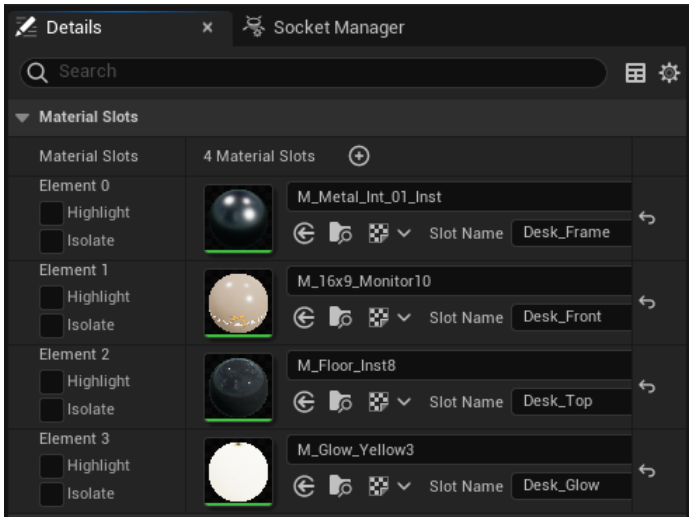
2. In the **Manage Media Targets** window, select the **Target Selected Material Textures Only** checkbox.
3. Then select the **Publish** checkbox for the **Texture Parameters** to which media can be applied and deselect the **Publish** checkbox for the **Texture Parameters** to which you don't want media applied.
4. Select **Apply** save your selection and close the window.

Naming Material Slots

Depending on how an actor is configured, it can have one or more material slots and any of those slots can be selected as a media target. The name assigned to each slot when the actor was designed may not identify each part of the mesh. It is good practice to rename the material slots so that you can easily identify the part of the mesh you want to use as a media target.

To change a material slot name:

1. In the **Outliner**, select the actor you want to use as a media target.
2. In the **Details** tab, scroll down to the **Static Mesh** section and double-click the **Static Mesh** icon to open the editor.
3. In the editor, in the **Material Slots** section, in the **Slot Name** fields, enter names that will make it evident which part of the actor is being referenced.



Static Mesh Editor - Rename Material Slots

4. Select **Save** and close the editor.
5. In the **Template Links Manager**, add the actor you've just edited, if it is not already listed.
6. In the **Media Target** column, select the actor and then select the **Edit** button.

   The new material slot names will appear in the **Material Slot** list.

   ★If there is a material slot that does not contain a texture parameter, it will not appear in the list, as it cannot be used as a media target.
7. In the **Manage Media Targets** window, select the **Target Selected Material Textures Only** checkbox.
8. Then select the material slot/texture parameter that you want to use as a media target.
9. Select **Apply** to save your selection and close the **Manage Media Targets** window.

Voyager and Lucid Studio

Lucid Studio is a studio operator control software developed by Ross Video. Studio operators should never need to go into the Unreal Editor to adjust elements in the virtual set. Instead, Lucid Studio communicates with an Unreal Engine plugin that parses all the objects in the scene, and interfaces the objects to its user interface.

The Lucid Studio Plugin is the interface between Lucid Studio and Voyager. When running Lucid Studio with the Voyager renderer, you need to set up communication between Lucid Studio and Voyager.

See Configuring the Lucid Studio Plugin for configuration instructions.

Lucid Studio Interface

What is Interfaced?

Currently Lucid Studio interfaces the following actors:

- Static Mesh Actors
- Lights
  - Point Lights
  - Spot Lights
  - Directional Lights
- Skeletal Mesh Actors
- Cameras
  - Camera Actors
  - Cinema Camera Actors
- Sequences and Matinees
- Floating Text

What can be controlled through Lucid Studio?

These are the basic properties that can be controlled through Lucid Studio:

- Position
- Scale
- Rotation
- Visibility
- Foreground/Background (in external compositing only)
- Materials
  - Lucid Studio can add video/textures from the UI to any static mesh object.
  - Lucid Studio can also change materials to other materials found in the scene.

  ★When using Voyager in **Game** mode, materials that haven't been assigned to an object will not be available in Lucid. It is best practice to create a few objects that are positioned beneath the floor in the scene, and assign all the materials you want to be able to access in the router to those objects, creating a "material library" for Lucid.
- Light color and intensity
- Text

Can Lucid Studio Interface Blueprints?

Lucid Studio can select and interact with (move, rotate, scale, show/hide, etc.) blueprint actors through its **Position** panel, but not with the blueprint coding. If a mesh is a component of a blueprint, Lucid Studio will not interface it. However, if you make the mesh a child of a Static Mesh Actor, Lucid Studio will be able to control it.

Lucid Studio can also interface with blueprints through the **Logic** panel. This requires that you select a Lucid Studio node in the blueprint and make it part of the **Exec** flow. In Lucid Studio, you will use the **Renderer Logic** function block to access the Voyager blueprint.

Naming convention

When you create an actor blueprint in Voyager that you want to control with Lucid Studio, you need to preface the name of the blueprint with "**BP**_", e.g., **BP_Lucid StudioActor**. This will ensure that Lucid Studio will see the blueprint. Once you place the actor in the scene, you can change the name in the **Details** tab, if you like. Lucid Studio will continue to see it as long as the source blueprint uses the correct naming convention.

For further information on using a Voyager blueprint with Lucid Studio, see the *Lucid Studio User Guide* (***Lucid Studio > Logic > Renderer***.)

BP_FreeRoaming_Camera

This actor is used to switch from the tracked Voyager Camera to another camera that will not use tracking data. This allows you to use Lucid Studio to trigger a trackless move like flying away from the talent in front of the green screen. If you created your project using either the VS Template or the AR+VS Template, there will already be a BP_FreeRoaming_Camera actor.

To switch from a tracked camera to a trackless camera:

1. In the **Outliner**, select the **BP_FreeRoaming_Cam** and double-click **Edit BP_FreeRoamingCam** to open the blueprint.
2. In the **Event Graph**, right-click in the blueprint and in the **Search** field, type `Trackless`.

   If you don't get any results, deselect the **Context Sensitive** checkbox.
3. From the results, select the **Set Trackless Camera** node.
4. In the **Set Trackless Camera** node:
   - Connect the **Target** pin to the **Operator** node.
   - Connect the **Trackless Camera** pin to the **Free Roaming Voyager Camera**.
   - In the **Blend Time** field, enter a value to set the transition time from the tracked camera to the untracked camera.
   - From the **Blend Func** drop-down, select the easing/smoothing method for the transition.
   - Use the **Blend Exp** field to enter a value to modify the **Blend Function**.

   For more information about the **Blend Time** and **Blend Func** settings, see the Unreal Engine documentation.
   - Select the **Lock Outgoing** checkbox to ensure that the transition goes to the targeted Trackless camera without resetting.

   OR
   - Leave the **Lock Outgoing** checkbox unchecked to have the focus returned to the tracked camera before switching to the targeted Trackless camera.

Adding a Free-Roaming Camera

You can add an additional Voyager camera (called the **BP_FreeRoaming_Cam**) that will not use tracking data. This allows you to switch between cameras, to get different viewpoints while you're working on your project. This is not something you can use while you're on air.

To add a Free Roaming Camera (optional):

- In the **Place Actors** tab, begin typing "bp" and from the results, drag the **BP_FreeRoaming_Cam** actor into the level.

Adding a Texture to Part of an Object (Legacy Method)

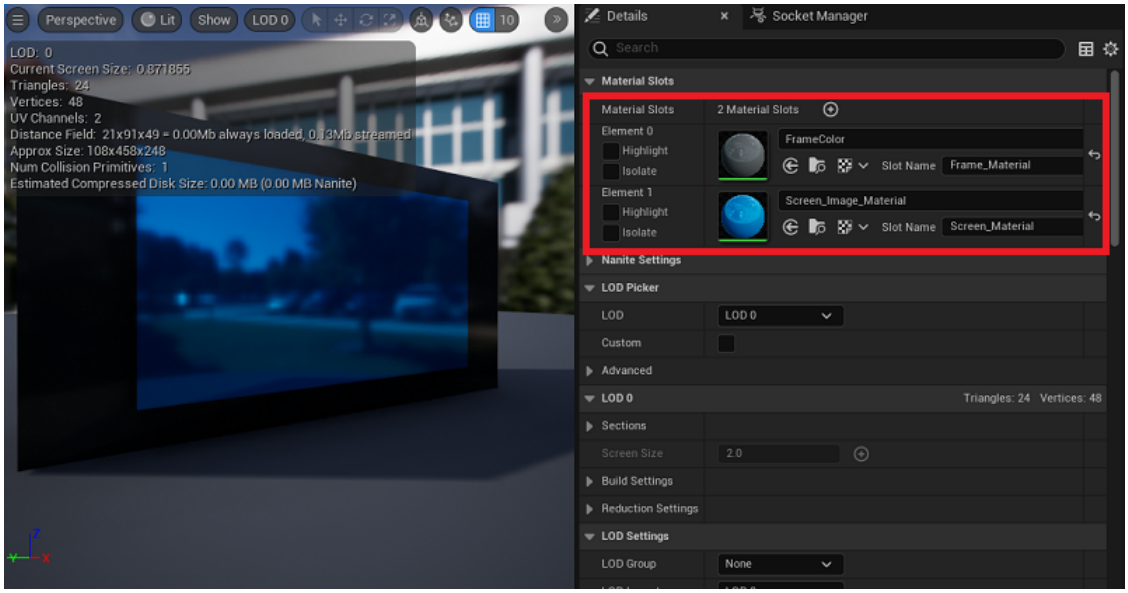
When you assign a video or texture to a static mesh object in your Voyager project, it's important to understand how to name the material slots, so that the texture is displayed correctly.

★This section applies to the legacy method previously used to make material slots visible in Lucid Studio. To learn about renaming material slots and accessing them through the **Template Links Manager**, see Targeting Selected Material Textures.

- If the selected Voyager object does not have any material slots with the **"TG_"** prefix, the video or texture will be applied to the entire mesh (all the material slots).
- If the selected Voyager object has one or more material slots with the **"TG_"** prefix on its name, the video or texture will be applied only to the material slot(s) that have that prefix.

To name the material slots of an object:

1. In your Voyager project, in the **Outliner**, select the object to which you want to add a video or texture.
2. In the **Details** tab, in the **Static Mesh** section, double-click on the **Static Mesh** icon to open the object for editing.



Static Mesh Material Editor

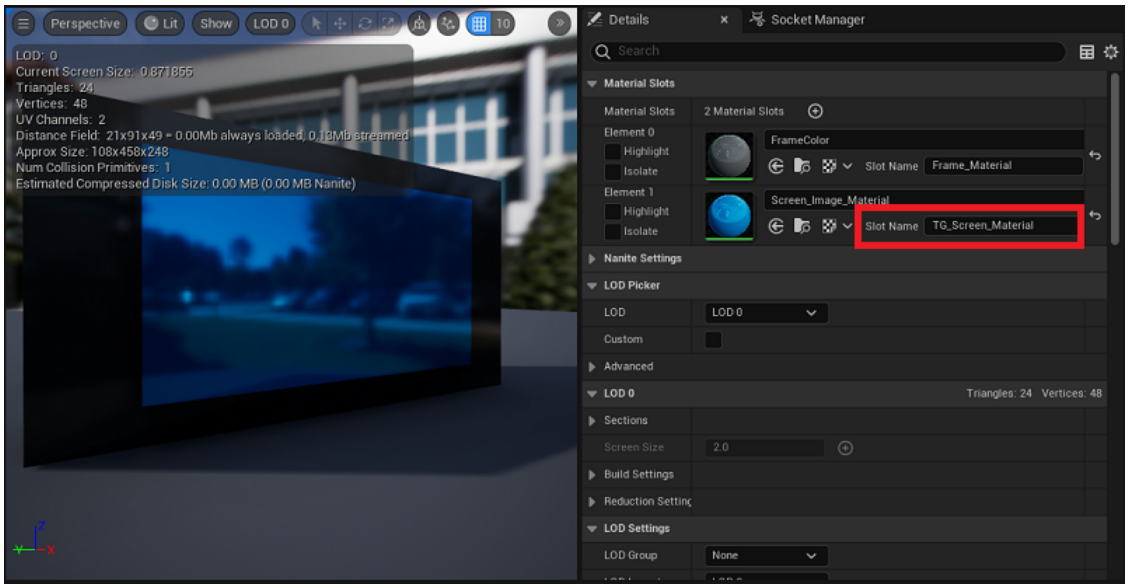
3. In the **Details** tab, in the **Material Slots** section, check the names of the material slots.

   By default, the material names do not have the **TG_** prefix.

If you applied an image to the static mesh object now, it would be applied to the entire mesh, as shown below:

Image Applied to Entire Static Mesh

4. Edit the name of the material slot to which you want to apply the source video or texture, to include the prefix **TG_**.



Static Mesh Material Editor - With Prefix Added

5. In the example, the video or texture should only be applied to the screen area, not the frame, so you would change the **Element 1 Slot Name** to **TG_Screen_Image_Material**.

   Subsequently, when a video or texture is applied to the target static mesh, it will appear only on the part of the mesh called **TG_Screen_Image_Material**, as shown below:

Image Applied to Selected Part of Mesh

Virtual Set Considerations

- Many visual issues only show up when looking through a tracked camera.

  There are ways to simulate a tracked camera through Lucid Studio, but mostly issues of flickering occur when the tracked camera is slightly defocused. This can be tested by running the level with camera tracking from Lucid Track applied.

- MipMaps are important.

  When using big textures for detailed background, use mipmaps. Backgrounds in virtual sets are typically a bit out of focus. If mipmaps are not used, details start to flicker.

  An example of textures that should be mipmapped are the images on the screens in the background:

  Using MipMaps

- Avoid thin lines.

  When the camera is slightly out of focus, thin lines can start to flicker because the temporal AA will blur the pixels completely with neighboring colors. Ensure that lines are not too thin from the expected camera positions.

- Avoid high emissive values.

  High emissive values cause flickering when zoomed out.

  The best way to test for flickering is to run the level in the studio. Alternatively, you can test it by running it through Lucid Studio.

  When running with a tracked camera, we will run a defocus filter as a post-process effect. In practice, cameras will send data slightly defocused. This defocus may cause flickering on screen if there are thin lines or high emissive values. If you cannot run it through a real TV studio test, run it through Lucid Studio.

- Screens must be seperate meshes from the TV. Screens should also be children of the TV.

  This is because Lucid Studio can only apply textures and materials on whole meshes.

- Screen UVs must match the mesh so that an arbitrary video in 16:9 format can be shown without being zoomed in or cut.

  When creating screen meshes make sure that the UVs do not crop out arbitrary textures. Do not create screen objects so that they only work with a single texture or video.

- Screens prefer emissive values and should probably be nonlit.

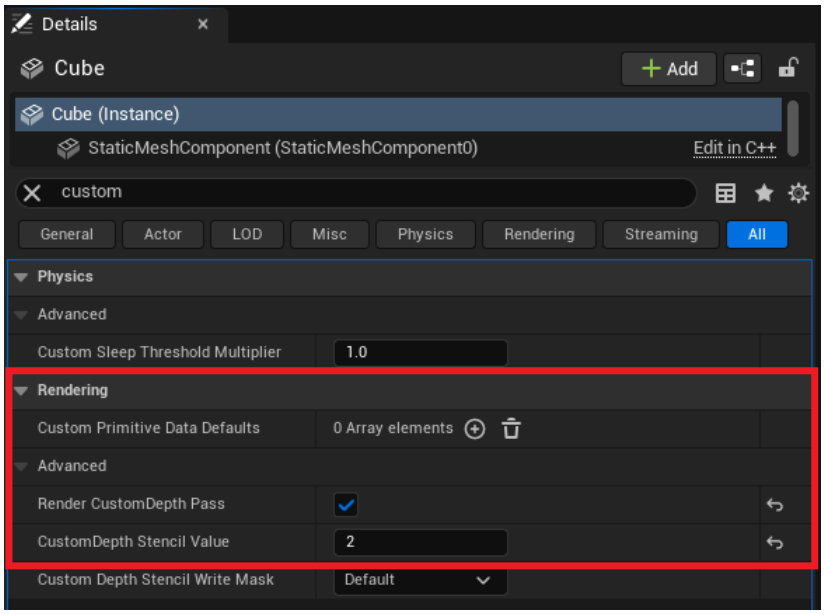
  Video on lit screens can be pretty performance intensive, and if the screen is already emissive, it does not need lights on it.

- In external composite virtual sets, do not set a translucent material on a foreground object, as this will allow the background to be seen through the object.

- In external compositing, you can configure an object or objects in your scene to always appear in the foreground or always appear in the background. If you have objects configured to always appear in the foreground, you need to also configure the background objects.

To enable an object to appear in the foreground:

1. In the **Outliner**, select the object you want to appear in the foreground.
2. In the **Search Details** field, type `custom` to get to the **Rendering** tab.



CustomDepth Settings - External Compositing

3. In the **Rendering** tab, select the **Render Custom Depth** checkbox.
4. In the **CustomDepth Stencil Value** field, enter **2**.
5. Then select the object(s) you want to remain in the background.
6. This time, in the **CustomDepth Stencil Value** field for each object, enter **1**.
7. Save your project.

   A custom depth value of 1 will automatically be applied to all objects without a custom depth value set, when you select **Update Project Settings**.

Retaining Existing Material Properties

Lucid Studio can change textures on materials and retain their properties, provided the material is built with special parameters.

When Lucid Studio sets a texture or video to a mesh, it typically applies a basic surface material with base color, nothing more, so you need to ensure that the existing material properties are retained.

To retain existing material properties:

1. In the **Content Browser > Materials**, double-click the material whose properties you want to retain, to open the **Material Editor**.

   You'll see the blueprint with all of the material parameters in the **Material Editor**.

2. In the **Material Editor**, add two **TextureSampleParameter2D** textures with the following names:

   - UX_Texture_A
   - UX_Texture_B

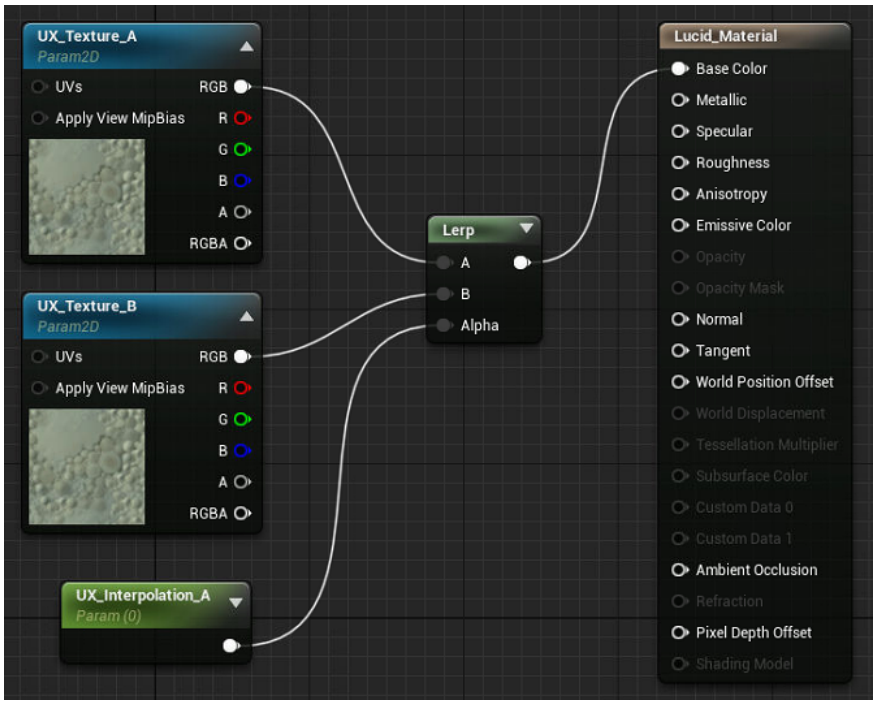
3. In the **Parameter Defaults** tab, make sure both parameters are set to the default texture.

4. Add a **ScalarParameter** with the name:

   UX_Interpolation_A

5. Add a **LinearInterpolate** (Lerp) node so that Lucid Studio can transition smoothly between two textures over time.

   It will look like this:



Setting up Lucid Studio Materials

Augmented Reality Set Considerations

This section provides some design recommendations to consider when creating an augmented reality set as they apply to either internal or external compositing.

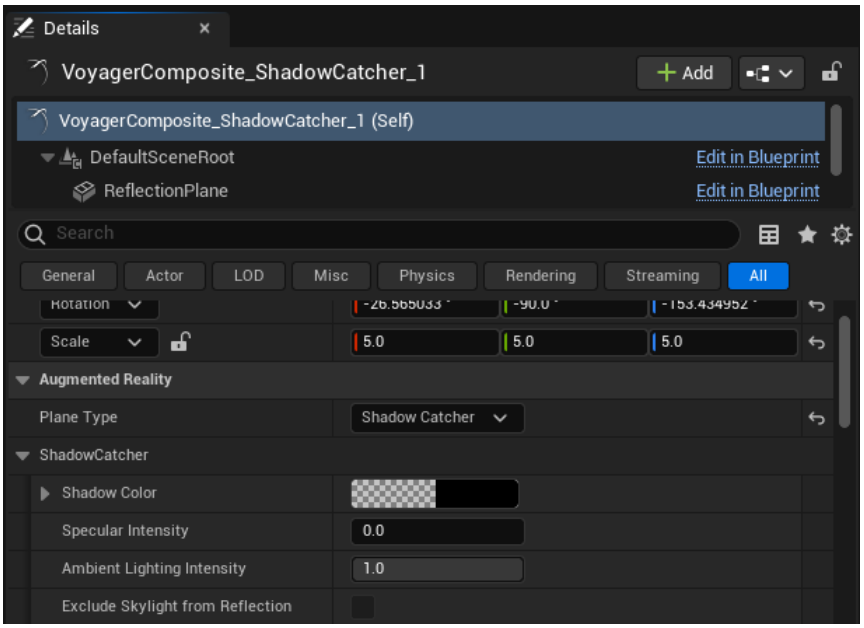
Internal Compositing

- You can use the **VoyagerComposite_ShadowCatcher** to display the reflections of virtual objects in the scene (for example, if they are sitting on a shiny surface), by setting the **Specular Intensity** to a value greater than **0** up to a maximum of **1**. However, if you are also using the **VoyagerComposite_Reflection** actor to reflect a part of the live background feed, you need to position the **Reflection** actor so that it is not reflecting on the **ShadowCatcher**.

  Otherwise, both the reflections of the virtual objects and the reflection of the live background feed could be displayed on the real world floor.

  You can adjust the **Ambient Lighting Intensity** to make the **ShadowCatcher** shadow better match the other shadows in the scene. A value of **0** disregards all the ambient lighting when generating shadows, while anything above **0** factors in ambient lighting, with **1** using all the ambient lighting.

  For more information on **Ambient Lighting** and **Ambient Occlusion**, see the Unreal Engine documentation.

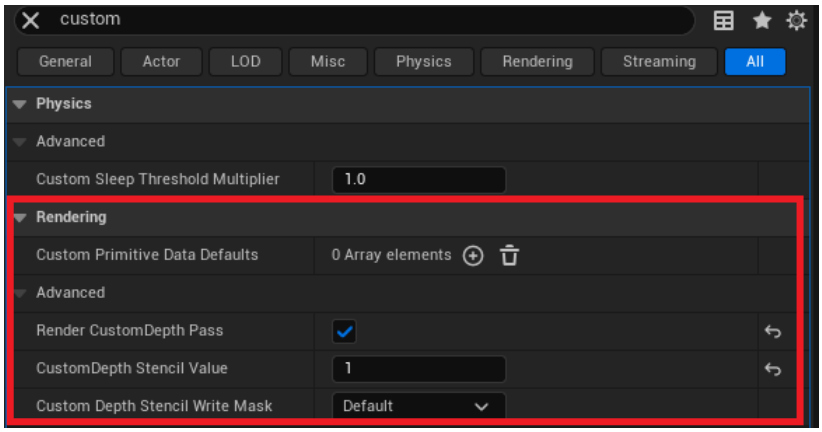


VoyagerComposite_ShadowCatcher

- In internal compositing, when you add a new object to the scene, you need to configure the **CustomDepth Stencil Value** to differentiate the new object from the rest of the scene.

To configure the CustomDepth Stencil Value:

1. In the **Outliner**, select the new object you've added.
2. In the **Search Details** field, type `custom` to get to the **Rendering** tab.



CustomDepth Settings - Internal Compositing

3. In the **Rendering** tab, select the **Render CustomDepth Pass** checkbox.
4. In the **CustomDepth Stencil Value** field, enter **1**.

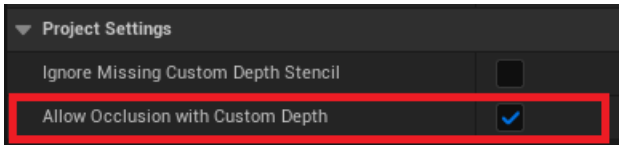
   This allows the object to be rendered on top of the composite plane and considered in the screen space reflection.

   A custom depth value of **1** will automatically be applied to all objects without a custom depth value set, when you select **Update Project Settings**.
5. Save your project.

   ★If you have a complex AR set with many virtual objects, many of which are hidden behind other objects, it is recommended that you turn on the **Allow Occlusion with Custom Depth** option.

To turn on the Allow Occlusion with Custom Depth option:

1. In the main toolbar, double-click the Voyager icon to open the **Voyager Operator** editor.
2. In the **Project Settings** section, select the **Allow Occlusion with Custom Depth** checkbox.



Voyager Operator - Allow Occlusion with Custom Depth

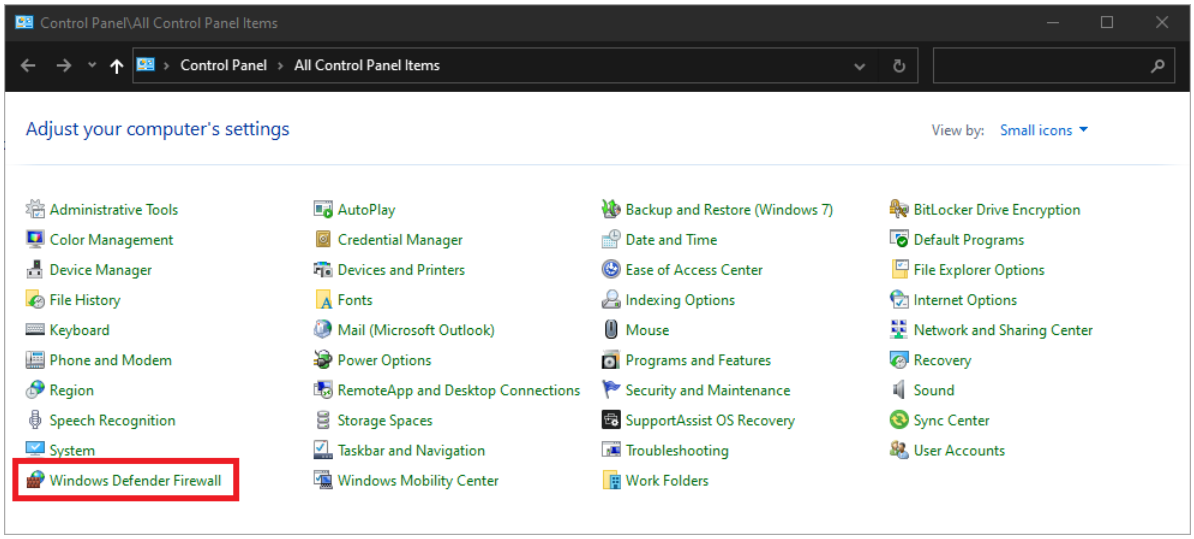
3. Select **Save** and close the editor.

Appendix A: Enabling a Port Number in the Firewall

When using Voyager, you need to make sure that any port you are using to listen to connections has been enabled in the Windows Defender Firewall.

To enable a port number:

1. In the **Control Panel**, select **Windows Defender Firewall**.

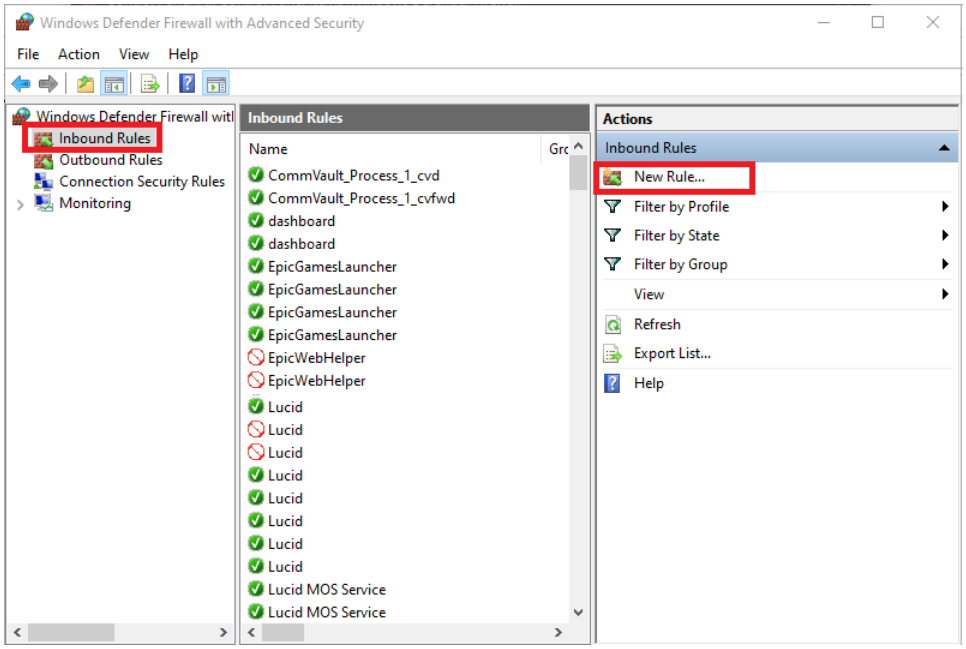


Control Panel

2. Select **Advanced settings > Inbound Rules**.

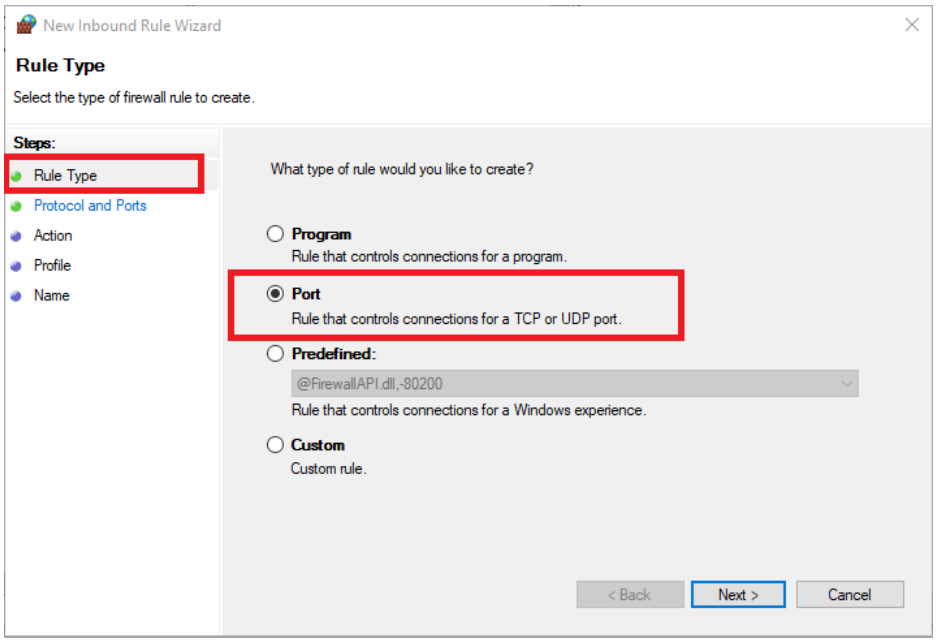
Advanced Settings

3. In the **Actions** pane, select **New Rule**.



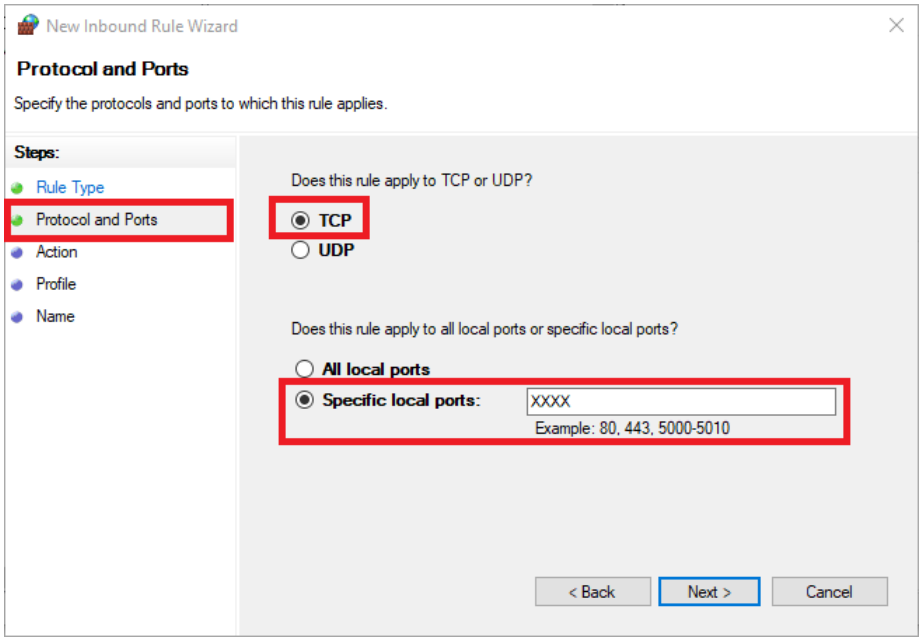
Actions - New Rule

4. In the **Rule Type** window, select **Port** and select **Next**.



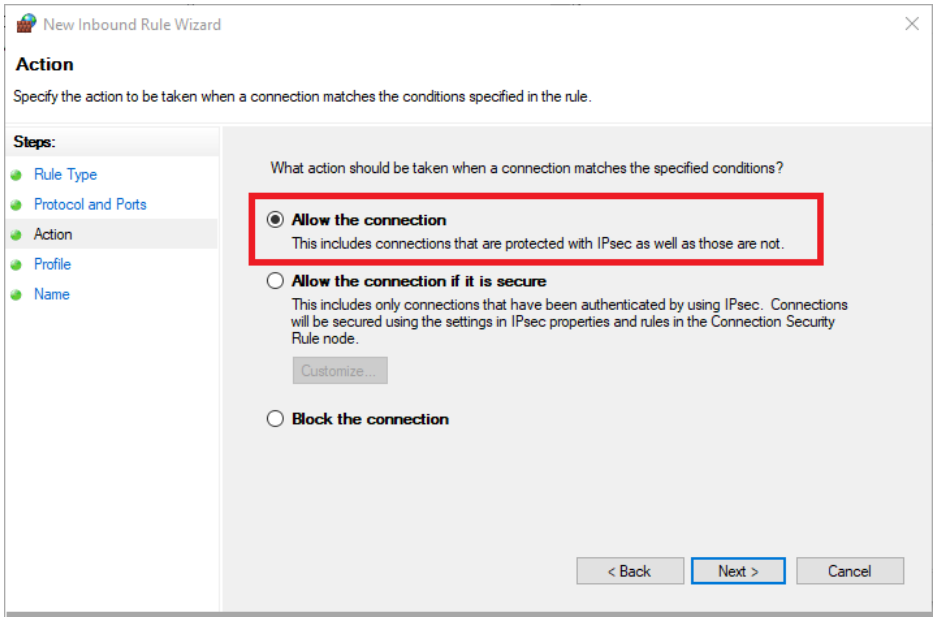
Rule Type - Port

5. In the **Protocol** and **Ports** window, select **TCP**.



Protocol and Ports

6. Then select **Specific local ports**, enter the port number you will be using and select **Next**.
7. In the **Action** window, select **Allow the connection** and select **Next** again.



Allow the Connection

8. In the **Name** window, enter a name for the new rule (e. g. Voyager Web API, Lucid Studio, RossTalk, etc.) and select **Finish**.
9. Close all the windows.

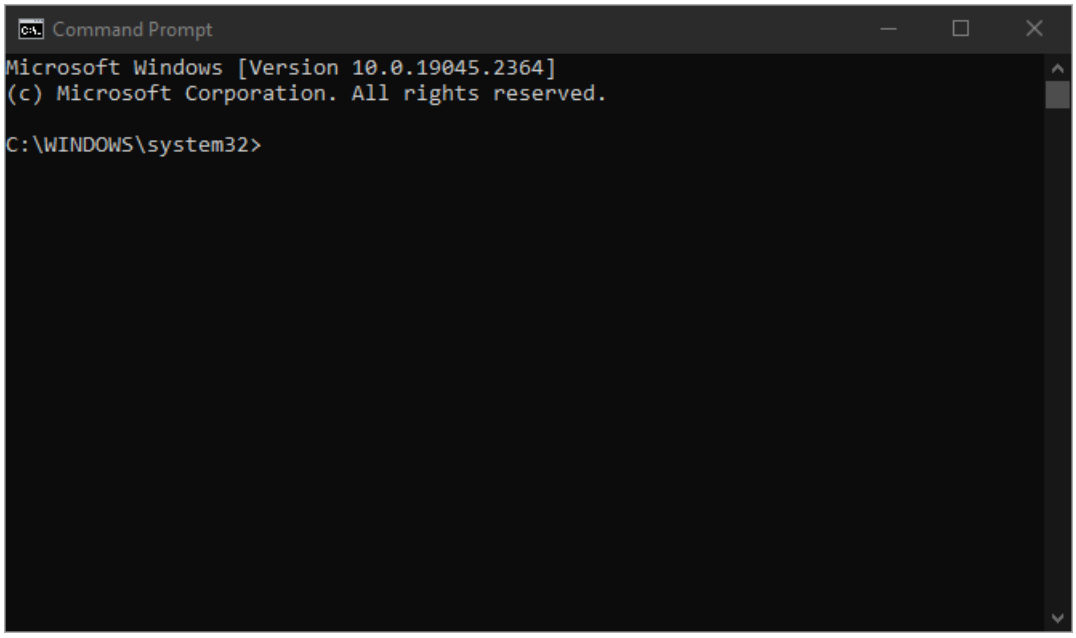
Appendix B: Testing SLP

If SLP is enabled in the Voyager Tracker and port 427 is also enabled, but services on your network aren't being discovered, you can test it using the following procedure.

To test SLP:

1. In the Windows **Search** field, start typing "`command prompt`".
2. In the results, select the **Command Prompt** application.

   The **Command Prompt** window opens.



SLP - Command Prompt Window

3. Type "`cd c:\program files\openslp`".
4. Then type "`slptool findsrvs service:myserv.x` " replacing "myserv.x" with the computer host name and company name..

   A list of connected services is returned.
5. Then type "`slptool findattrs service:myserv.x://myhost.com`", replacing "myserv.x://myhost.com" with one of the services that was returned in the previous step.

   A list of attributes for the service is returned. Attributes include IP addresses, port number, version number and engine number of the service.
6. If you don't get any results, contact Technical Support for assistance.

Appendix C: Troubleshooting

System

The following system setting changes are recommended for optimal performance.

CPU

Hyper threading should be disabled in the BIOS setup,

nVidia

Use the recommended nVidia Driver version: Studio WHQL Quadro Certified 581.80.

Make sure you don’t have **nVidia Geforce Experience** installed. Uninstall it if installed.

Power management mode should be set to **Prefer maximum performance**.

To set power management mode:

1. Right-click on the desktop and select **nVidia Control Panel**.
2. In **3D Settings > Manage 3D Settings**, select the **Global Settings** tab.
3. In the **Settings** box, scroll down to **Power management mode** and select **Prefer maximum performance**.

Microsoft Windows

The Windows 10 March 2023 Update (KB5023696) for version 22H2 may affect the performance of the engine.

Virtualization Based Security may affect the performance of the engine.

Disable Virtualization Based Security if it is enabled.

To disable Virtualization Based Security:

1. In the **Windows Security** application, go to **Device Security > Core Isolation > Core Isolation Details** and set **Memory integrity** to **Off**.
2. In the **Registry Editor**, go to **HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\DeviceGuard** and set **EnableVirtualizationBasedSecurity** to **0**.

   If this setting doesn't exist, then Virtualized Based Security is not enabled.

Unreal Engine

In Voyager, change the following settings:

Anti-Aliasing Method

Temporal Anti-Aliasing (TAA) should be used.

★Do not use Temporal Super-Resolution (TSR).

To select Temporal Anti-Aliasing (TAA):

- In the **Edit** menu, go to **Project Settings > Engine > Rendering > Default Settings** and from the **Anti-Aliasing Method** drop-down, select **Temporal Anti-Aliasing (TAA)**.

Lumen

Lumen (Unreal 5) has an important performance impact. Disable Lumen if it is not strictly required.

When Lumen is enabled, **Generate Mesh Distance Fields** is also enabled. If you disable Lumen, you should also disable **Generate Mesh Distance Fields**.

To disable Lumen:

- In the **Edit** menu, go to **Project Settings > Engine > Rendering > Global Illumination** and from the **Dynamic Global Illumination Method** drop-down, select **None**.
- In the **Edit** menu, go to **Project Settings > Engine > Rendering > Reflections** and from the **Reflection Method** drop-down, select **None** if you don't need reflections at all, or select **Screen Space** if Unreal Engine's 4 screen space reflections are good enough.

Nanite

Nanite (Unreal Engine 5) should generally be enabled wherever possible. Any static mesh that has it enabled will typically render faster and take up less memory and disk space..

To enable Nanite:

- Go to **Edit > Project Settings > Engine > Rendering** and select the **Nanite** checkbox.
- When importing new meshes, select **Build > Build Nanite** (it might be disabled by default for some meshes).
- For previously imported meshes, enable Nanite (**right-click > Nanite > Nanite**).

Ray Tracing

Ray Tracing is expensive; enable it only if strictly necessary.

To disable Ray Tracing:

- Go to **Edit > Project Settings > Engine > Rendering > Hardware Ray Tracing** and deselect the following checkboxes:
  - **Support Hardware Ray Tracing**
  - **Ray Traced Shadows**
  - **Ray Traced Skylight**

Virtual Shadow Maps

Virtual Shadow Maps is Beta and may have some performance impact. Use **Shadow Maps** instead.

To use Shadow Maps:

- Go to **Edit > Project Settings > Engine > Rendering > Shadows > Shadow Map Method** and from the drop-down, select **Shadow Maps**.

DirectX 12

Enable DirectX 12 only if you need Ray Tracing or another Unreal Engine 5 feature that requires it.

To disable DirectX 12:

- Go to **Edit > Project Settings > Platforms > Windows > Targeted RHIs** and from the **Default RHI** drop-down, select **DirectX 11**.

VLED Checklist

nVidia

For best performance, change the **Global Presets** and **Vertical Sync** settings as follows:

To change the Global Presets setting:

1. Right-click on the desktop and select **nVidia Control Panel**.
2. In **3D Settings > Manage 3D Settings**, select the **Global Settings** tab.
3. In the **Settings** box, scroll down to **Global Presets** and from the drop-down, select **Workstation App Dynamic Streaming**.

To change the Vertical Sync setting:

1. Right-click on the desktop and select **nVidia Control Panel**.
2. In **3D Settings > Manage 3D Settings**, select the **Global Settings** tab.
3. In the **Settings** box, scroll down to **Vertical Sync** and from the drop-down, select **Use the 3D Application setting**.

Network

Check that the network adapter and hardware meet the following requirements:

- Verify the Windows network adapter is running at 10 Gbps.
- Verify you have the proper 10 GB network hardware:
  - — 10 GB Ethernet Adapter (Voyager's Intel x550 network adapters support 10 GB.)
  - — Factory 10 GB Ethernet cable, ideally not DIY cables.
  - — Use a good 10 GB Ethernet switch.

Disable Power Management for the Network Adapter.

To disable Power Management:

1. In **Network and Internet settings > Advanced network settings > Change adapter options**, double-click your connection to open the **Wi-Fi Status** window.
2. Select **Properties** and in the **Networking** tab, select **Configure**.
3. Select the **Power Management** tab and clear the **Allow the computer to turn off this device to save power** checkbox.

Switchboard

Fullscreen Optimization

Disable Fullscreen Optimization on every Unreal Engine node executable.

To disable fullscreen optimization:

1. Go to **C:\Program Files\Voyager\Engine\Binaries**.
2. Right-click on the **UnrealEditor.exe** file and select **Properties**.
3. In the **Compatibility** tab, in the **Settings** section, select the **Disable fullscreen optimizations** checkbox.

   OR

   Launch Voyager Switchboard and in the Voyager Switchboard editor, select the **Fix ExeFlags** button.

Internet Connection

An Internet connection may be required for first execution.

The first time you run Voyager Switchboard on a system, it may download some dependencies. Make sure the machine is connected to the Internet before running Voyager Switchboard the first time.

Project Path and Name

Verify the project path and file name.

- Project name should not include spaces or special characters.
- Project name should have a maximum of 20 characters long.
- Project path should not have spaces or special characters.

Switchboard Listener

Voyager Switchboard Listener needs to be running on all the nodes.

Licenses

Verify your machines have the required licenses. Voyager Switchboard may open and close abruptly with no prompt if no valid licenses are found. Check the log files to verify that you have the appropriate licenses.

- The nDisplay Master node requires the Voyager nDisplay Controller License.
- The nDisplay nodes require the Voyager nDisplay Node License.

Projects

Run your projects on all machines - to check for any problems running.

With nDisplay if any Node times out (popup waiting for answer) then the whole nDisplay cluster will quit. You will just see a Black Screen

nVidia Sync

Use CAT-5 (or higher) factory manufactured Ethernet cables for connecting the nVidia Quadro Sync cards.

Do not use Ethernet hubs or switches to branch or extend the signal, as it is not TCP/IP.

nDisplay Synchronization Policy should be set to nVidia Sync.

To set nDisplay Synchronization Policy to nVidia Sync:

1. In your Voyager project, in the **Outliner**, select the **VLEDStage** actor to open the editor.
2. In the editor, in **Details > Configuration > Cluster**, expand **Render Sync Policy**.
3. From the **Type** drop-down, select **Nvidia**.

   OR

   In Voyager Switchboard Launcher, select **Settings > Settings > nDisplay** and from the **Render Sync Policy** drop-down, select **Nvidia**.

   ★The setting chosen in Voyager Switchboard Launcher over-rides the setting chosen in the Voyager project.

VLED + Set Extension

Make sure the **Primary Device Used as Set Extension** checkbox is selected in Voyager Switchboard.

In the Voyager Switchboard Launcher, in **Settings > nDisplay Settings**, select the **Primary Device Used As Set Extension** checkbox.

Set Extension AR Voyager machine will be black if this checkbox is not selected.

Contact Us

Contact our friendly and professional support representatives for the following:

- Name and address of your local dealer
- Product information and pricing
- Technical support
- Upcoming trade show information

Technical Support	**Telephone:**	+1 613-686-1557 +1 833-859-0499 (Toll free within North America) +800 3540 3545 (Toll free International) 1300 007 677 (Australia/Sydney)* *If the local support specialist is not available, your call will be transferred automatically to our North America center.
	Email:	techsupport@rossvideo.com

General Information	**Telephone:**	+1 613-228-0688 +1 844-652-0645
	Fax:	+1 613-652-4425
	Email:	solutions@rossvideo.com
	Website:	http://www.rossvideo.com

Visit Us

Visit our website for:

- Company information and news
- Related products and full product lines
- Online catalog
- Testimonials